Prepared with the Support of the
National Aeronautics and Space Administration

NASA Grant NSG - 8047

Grain Quality Inspection System

by

Clifford A. Flood, Jr.

and

David P. Singleton

Department of Agricultural Engineering
Auburn University
Auburn, Alabama 36849

and

Sidney N. James

Department of Electrical Engineering
Auburn University
Auburn, Alabama 36849

Final Report

Covering the Period January 23, 1977 to September 30, 1979

(NASA-CR-163682) GRAIN QUALITY INSPECTION SYSTEM Final Report, 23 Jan. 1977 - 30 Sep. 1979 (Auburn Univ.) 167 p HC A08/MF A01 CSCL 14D

N81-10445

Unclas

G3/38 29110

CONTENTS

List of Tables

List of Figures

SUMMARY AND CONCLUSIONS

An extensive review of grain quality indicators and measurement methods was conducted in order to assess the feasibility of using remote sensing technology to develop a continuous monitoring system for use during grain transfer operations. It was found that there is a large number of indicators of grain quality. Many have meaning or are of value only for a small segment of the user population but are very important to the interested user. Most detection methods were found to be too slow or too expensive to be incorporated into the normal inspection procedure of a grain elevator on a continuous basis. Present sensor technology is such that, at the present, remote sensing of most quality indicators would be both difficult and impractical.

Two indicators were identified which showed potential for automation, however. Moisture content and Broken Corn and Foreign Material both show potential for automation and are of an economic value which will make an automated system of value to a grain handler.

A microprocessor based system which utilizes commercially available electronic moisture meter was developed and tested. The system will control sampling and measuring of moisture content at time intervals determined by the user. The system can also calculate running averages, send messages when upper and lower limits are exceeded, and perform other control functions. The system when tested performed all functions expected of it.

A method for automating BCFM measurement is described. Incorporation of BCFM measurement into the microprocessor based system tested can be easily accomplished with the development of some rather simple mechanical equipment.

A complete system description is presented along with performance test results.

INTRODUCTION

At each point in the market system where grain ownershop changes, there is need for a determination of the quality of that grain. This is necessary in order to establish a price and also to assure that the material is suitable for the intended use of the buyer.

The volume of grain exported by the U.S. in recent years has made a major contribution toward reducing our trade deficit. Complaints of poor grain quality are detrimental to the expansion of this export trade. Improved methods of evaluating grain quality could result in an increase in the quality of grain shipped as well as better acceptance of U.S. grains on both the domestic and world markets.

Regardless of the size of a shipment, grain quality assessment is currently performed by individual inspectors. Samples are obtained manually with the use of probes. Evaluation of a sample large enough to be representative of large bulks of grain can be both time consuming and tedious. An automatic sampling and sensing system should improve the reliability, speed and ease of quality assessment.

This project addressed the problem of automating the quality evaluation process within a bulk grain handling system. The summary objective was to improve grain quality inspection through utilization of existing technologies and equipment. Work was carried out in two phases. The first phase was an evaluation of automation potential for corn and soybeans. Specific objectives were:

1. To identify indicators of grain quality for corn and soybeans.
2. To determine which indicators can be evaluated using an automated remote sensing system.
3. To develop a systems concept for implementation of automated quality sensing for corn and/or soybeans.

The second phase of the project involved the design of selected sub-systems of an automated inspection system for corn. Specific objectives were:

1. To develop an automated moisture content measurement system for corn.
2. To investigate methods for automated evaluation of BCFM.
3. To design a control and data handling unit for an automated corn quality sensing system.

PRINCIPLE FINDINGS

A. Quality Indicators and Measurement Methods.

An extensive and detailed literature review was undertaken to examine past and present research related to this project. The review was to serve two purposes, identify grain quality indicators and reveal methods measuring these quality indicators.

Appropriate books and journals were examined and applicable information was noted. Books pertaining to all phases of grain harvesting, handling, processing and grading were of interest. Manuals explaining federal regulations concerning grain grading gave guidelines for quality analysis of grain. Agricultural and chemical journals involving grain quality analysis were reviewed and articles of interest were examined closely and those found to be of value noted. Abstracts from other scientific journals provided more information. Journals were reviewed for eight to ten years into the past and further if articles of interest were referenced in earlier issues.

In addition to this available written review several computer data bases were utilized to check for any material that had been missed or was unavailable from local sources. The computer stored data bases contain citations from journals in the area covered by the data base. Three data bases that covered our area of interest were chosen to be searched. Biosis Previews contained 1,700,000 citations, Agricola contained 960,000 citations and CRIS contained reports on 30,000 projects. The Biosis Previews and Agricola contained citations from 1970 until the present. CRIS (Current Research Information System) contained reports on agricultural research in progress from 1974 until the present. All titles that used pertinent key words were printed on computer printout. Then articles

of value were obtained and those of importance were retained.

Visits at several locations were made so we could meet other researchers with expertise in grain research. Sites visited were Purdue University, Ohio State University, University of Illinois, U. S. D. A. Instrumentation Laboratory at Beltsville, Cargill Inc. Grain Research Lab, and a grain quality conference were attended. We were able to view equipment used in commercial operation. Experimental equipment which may be used in quality sensing operations was identified. Individuals consulted provided useful information concerning important quality indicators and problems involved with sensing these quality indicators.

The identity of an important grain quality indicator depended on the particular use intended for the grain. Different uses required grain of highly different quality characteristics. Among the quality indicators, of course, were the grade factors from the U.S. Standards for Grain handbook. Other parameters that would indicate quality are odor, stress and strain cracks, presence of aflatoxin, various localized insect or disease infestations, and content of economically valuable compounds such as proteins, fats, oils and starch.

Test weight per bushel of the grain generally indicates higher quality with higher test weight if all other factors are equal. A dry miller is the only processor who considers high test weight extremely important. Other processors are not too worried about test weight if it is within a reasonable limit of normal. Test weight is measured by taking a known volume, usually a quart, and weighing the sample on a special scale that gives a value in lbs. per bushel.

Moisture content is important as wet weight of grain varies with moisture content and grain is sold on a wet weight basis. Moisture content above certain levels also decreases keeping quality of grain. The only

proven exact method of measuring moisture content is oven drying. Moisture meters provide a quicker method which yields good results. A recently developed method is Infrared Reflectance Spectroscopy which yields values comparable with moisture meters in accuracy.

Broken corn, splits in soybeans, and foreign material all affect quality of grain. The foreign material lowers the nutritional value available in the grain. The other factors provide a substrate for microbial growth and potential development of deleterious metabolic by-products. Measurement is accomplished by the use of sieves or with a dockage tester.

Amount of heat damaged corn and other surface and sub-surface damage affects the appearance of grain as well as providing an area of growth for microbes. If the percentage of damage is kept low it has very little affect on nutritional value. This type of damage is detected by personal examination of a sample so exact measurement of damage is time consuming.

Odor and insect or disease infestations give an indication of the past handling of the grain and the future keeping value. Insect or disease infestations is determined by personal examination. Odor is a very relative quality dependent on the individual. More exact determinations can be made by some forms of gas chromatography but this requires complex equipment, trained personnel, and is expensive and slow.

Stress and strain cracks indicate weakness in the grain and the development of increased breakage in the grain. They can be found by human detection and there is a colorimetric method but the values obtained are very relative.

Detection of aflatoxin is critical during years when it is prevalent since it is fatal to animals above certain limits. Blue light fluorescence may give an indication of the presence of aflatoxin but the only exact method is by gas chromatography:

Proteins, fats, oil and starch are the valuable components of grain so it would be useful to know what percentage of each a grain contained. Protein and oil can be measured by Infrared Reflectance Spectroscopy with a reasonable degree of accuracy, but fats and starch are measured by slow chemical extraction methods.

In summary, there is a large number of indicators of grain quality. Many have meaning or are of value only to a small segment of the user population. Usually they are very important to the interested user, however. Most detection methods are too slow or too expensive to be incorporated into the normal inspection procedure of a grain elevator on a continous basis. The potential for implementation of remote sensing of most quality indicators would be difficult and impractical with the present sensory technology. However, some of the indicators show potential for automation. Two indicators of high economic importance that show potential are moisture content, and Broken Corn and Foreign Material. These two indicators are of interest because their measurement techniques show more potential adaptability to automation and their economic value will make an automated system of value to a grain elevator.

The system that we foresee being developed is one where samples would be automatically pulled from the grain stream. Equipment for measuring moisture content, and broken corn and foreign material would process the samples. A central data processing unit then would take the indicator values and provide a hard copy record of the grain sample and its properties for future reference if needed.

B. Methods for Automated Evaluation of BCFM

During the project methods for automatically evaluating Broken Corn and Foreign Material (BCFM) in a corn shipment were investigated. This included looking at the present method of determing BCFM and examining research that had been done concerning alternative methods for measuring BCFM. These procedures were considered for their feasibility for incorporation into an automatic monitoring system.

The Grain Inspection Manual published by the Federal Grain Inspection Service defines Broken Corn and Foreign Material as kernels and pieces of kernels of corn and all matter other than corn which will pass readily through a 12/64 inch sieve, and all matter other than corn which remains in the sieved sample. These Standards also state that the 12/64 inch sieve shall be an aluminum sieve 0.0319 inch thick perforated with round holes 0.1875 (12/64) inch in diameter which are ¼ inch from center to center. The perforations of each row shall be staggered in relation to the adjacent row. The prescribed procedure is to take a representative portion ranging in size from 1 1/8 to 1¼ quarts cut from the original sample and measure BCFM using an approved type of dockage tester using the approved sieve. Approved dockage testers are the Carter Dockage Tester, Federal Dockage Tester, and the Emerson Kicker. Broken Corn and Foreign Material will consist of all material which passed through the sieve, and all other matter other than corn removed by hand from the mechanically cleaned corn. Use of the sieve to separate BCFM shows possibilities for being incorporated into an automatic monitoring system.

There have been suggestions for changing the designation BCFM. One idea is to let the quality parameter be designated as screenings which would include materials passing readily through a 4.762 mm (12/64 inch)

round-hole sieve and include peices of corn and other non-corn materials. There would be a separate measurement for dockage which would include any material other than whole or unbroken kernels of corn that remains in the sample after the removal of screenings. Dockage includes, but is not limited to, other grains, weed seeds, and pieces of cob or stalk. Another suggested change would be to use two sieves - 15/64 inch and 8/64 inch. Material that passes through the 15/64 inch sieve but not through the 8/64 inch sieve would be called Broken Corn. Material passing through the 8/64 inch sieve would be called Screenings. Foreign Material would be defined as described in the previous suggestion. Both of these ideas were suggested by researchers at the Proceedings of the 1977 Corn Quality Conference. Measurement of Screenings or Broken Corn with sieves shows possibilities for incorporation into an automated system.

Other suggestions by researchers for determining broken corn or damaged corn included use of a corn breakage tester, colorimetric methods, and photoelectric methods. None of these methods have any correlation between their results for corn breakage and damage, and the quality parameters used by the Federal Grain Standards. These methods are slow, taking from three to five minutes or longer, and require manual operation that would be hard to eliminate. As a result, these methods show little adaptability for use in an automated system.

Using a sieve for measuring BCFM as described by the Grain Inspection Manual shows some potential for being used in an automatic monitoring system. Consequently, the same method could be used for measuring Screenings on Broken Corn if either of those suggested quality parameters were adopted. Measurement of foreign material in either grading system would still have to be done by hand separation. A suggested system in an attempt to auto-

matically measure Broken Corn would be to pull a sample, weigh it electronically, pass it over the sieve, remove the corn that was retained on top of the sieve and electronically weigh the sample. Figure 1 contains a block diagram of a possible configuration of a Broken Corn monitoring system. The percent of Broken Corn could be calculated by the equation:

$$\%\text{ of Broken Corn} = \left(\frac{\text{wt. of original sample-wt. of retained corn}}{\text{wt. of original sample}}\right) \times 100 \qquad (1)$$

Calculation and control ability is already contained in the automatic moisture content monitoring system developed during this project. If the physical hardware could be developed to do the sieving and weighing, a system to monitor both moisture content and broken corn could be developed using the present system with some software revision and additional subroutines.

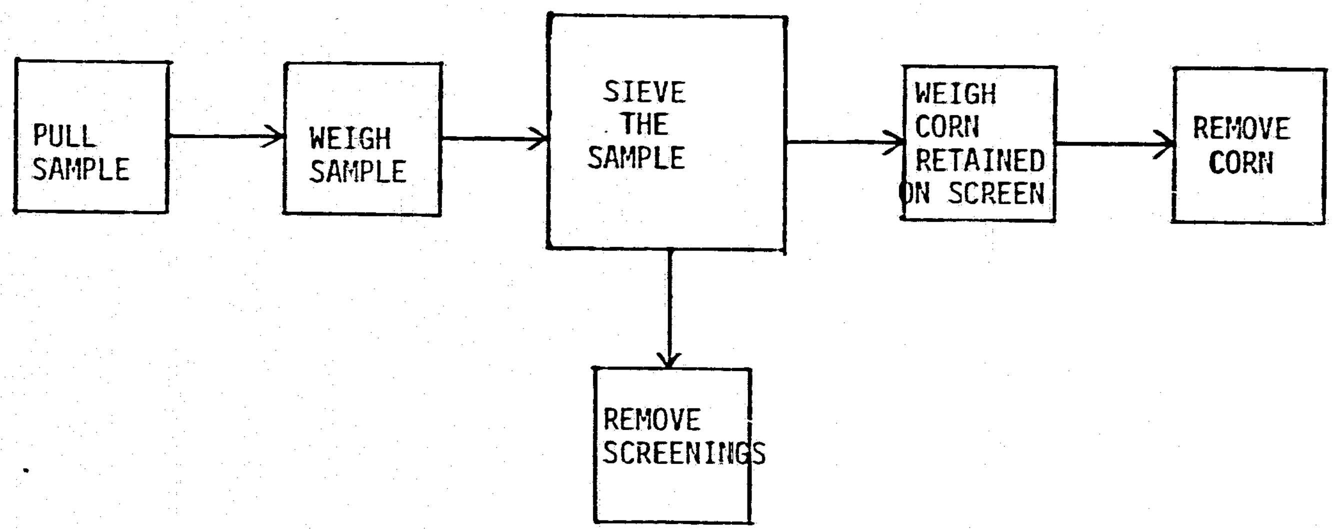


Figure 1. Block diagram of broken corn monitoring system.

C. System Implementation

A microprocessor based system was developed which can be used to automatically monitor the moisture content of grain as it is being transferred at an elevator. The system can also provide control functions and monitor any other quality parameter for which rapid measuring sensors are available. System development and implementation involved hardware design and construction, software development and performance testing.

Hardware Design and Construction

To do an effective job of monitoring the moisture content of grain being loaded or unloaded the system must fulfill certain basic requirements. The system must be able to control the moisture sensing operation and take the moisture content reading of a grain sample at periodic time intervals. After taking a moisture content reading mathematical operations must be performed to obtain an average moisture content value. Then a record output is required to show the moisture content reading, and average moisture content value. Also communication is required between system and operator to program the system for the type of monitoring operation to be performed, to provide system parameters and to show how monitoring operation is progressing.

A microprocessor is used to control the operation of the system and to provide mathematical capabilities with the addition of a calculator card. The decision making capabilities of the microprocessor are used to determine if moisture contents meet the desired criteria and what path the monitoring operation will follow. Input ports are used to interface with the moisture sensor. And a 20 mA circuit loop

on the microprocessor interfaces with the keyboard and printer to provide communication.

The moisture sensor used was a Burrows Model 700 Digital Moisture Computer. This moisture meter measures moisture content by measuring the capacitance of a 250 gram grain sample. A Binary Coded Decimal output is used to interface the moisture meter with the microprocessor.

Communication between operator and microprocessor is provided by a DEC-Writer II. It has a keyboard for typing in inputs to microprocessor and a printer for a paper printout of the record.

A block diagram of the system configuration and components is shown in Figure 2.

<u>Microprocessor card rack components</u>. - The microprocessor system used is a PRO-LOG CRS-81 Expandable 8080A/1702A Card Rack System. This is a card rack with space for sixteen cards of which ten card spaces are prewired. Provided with the system are an 8811A Processor card, 8116 ROM card, 8114 Input card and a 8115-1 Output card. Additional cards purchased to go into the card rack were two more 8116 ROM cards, an 8117 RAM card and a 8407 Serial interface card. Card spaces for these additional cards were prewired. The 8117 RAM card was used during system development but isn't required for the moisture monitoring system. Wiring schematics for these cards are shown in Appendix A. The card rack and cards are shown in Figure 3.

An eight bit 8080A microprocessor is on the 8811A processor card. The 8080A microprocessor is a 1 microsecond time state processor. Also contained on this card is 1024 bytes of eight bit 2102-4 Random Access Memory (RAM) which supplies the RAM requirement for the moisture monitoring system. Addresses for this RAM is from 3000 to 33FF. This card has one interrupt request line which is used for timekeeping for one minute sampling rate.

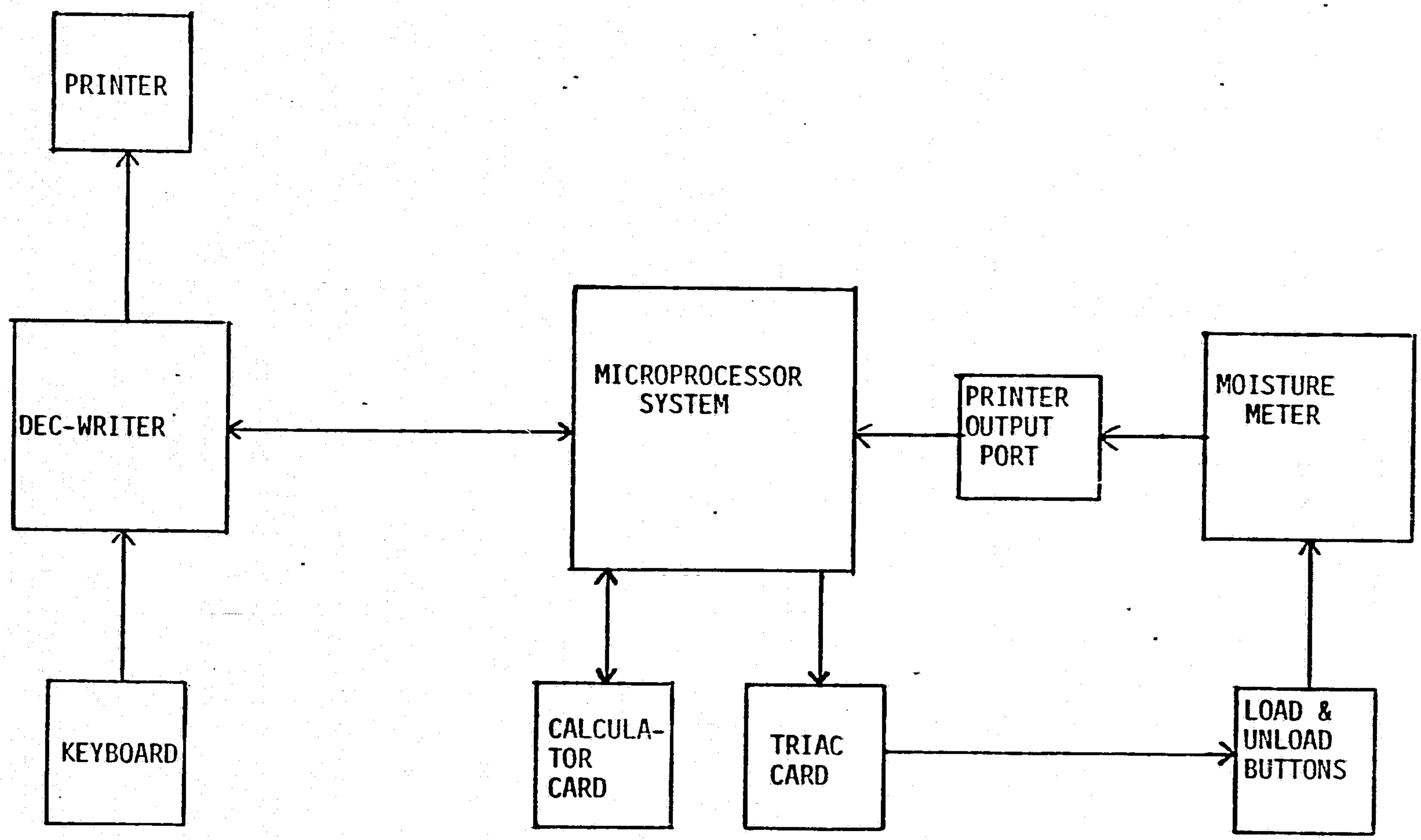


Figure 2. Block diagram of the automated moisture content monitoring system configuration and components.

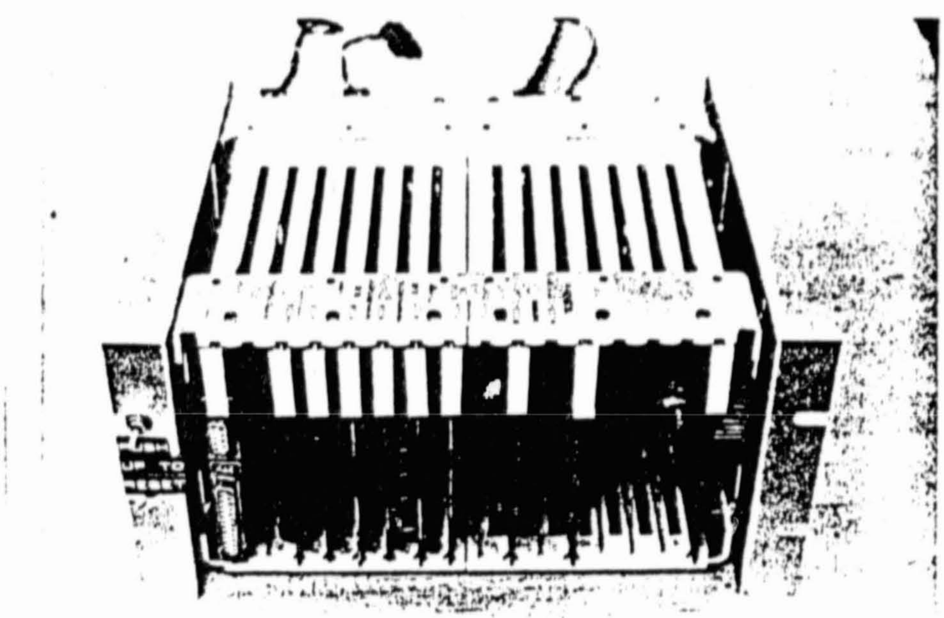


Figure 3. Card rack with cards inserted.

The 8116 ROM card contains space for 2048 bytes of eight bit 1702A Read Only Memory (ROM). The three 8116 ROM cards used in the prewired slots provide 6144 bytes of memory space. Addresses for this memory is 0000 to 17FF. The 1702A Erasible Programmable Read Only Memory (EPROM) can be erased by an ultraviolet light and programmed again with the M900 Programmer with PM9001A Personality Module for this type of EPROM. The programmer is shown in Figure 4.

Four eight-bit input selectors on the 8114 Input card provides four ports of eight input lines each for a total of thirty two input lines. The addresses of the four ports are 00 through 03.. Two input ports are used to bring in moisture content reading from moisture meter and one port is used for interfacing with 90006 Calculator card. Interface wiring is discussed in the sections pertaining to the moisture meter and calculator card.

The 8115-1 Output card supplied with the Card Rack System has four output ports with eight output lines each for a total of thirty two output lines. The address of the four ports are 00 through 03. These addresses are selected only when an OUT instruction is executed. One output port plus another output line is used to interface with 90006 calculator card and two lines from another output port are connected to the 8404-4 Triac card to operate solenoids mounted on moisture meter. Interface wiring for these ports are discussed in the calculator card and moisture meter sections.

An 8407 Serial Interface card was used to interface between microprocessor and DEC-Writer. This card has both RS-232 and TTY serial data communications lines. The TTY interface with 20 milliamp current loop was used since this was the type of interface on the DEC-Writer. The DEC-Writer II that was used had a 20 milliamp option with a connector which plugged into the 20 milliamp connector on the 8407 Serial Interface card as shown in Figure 5.

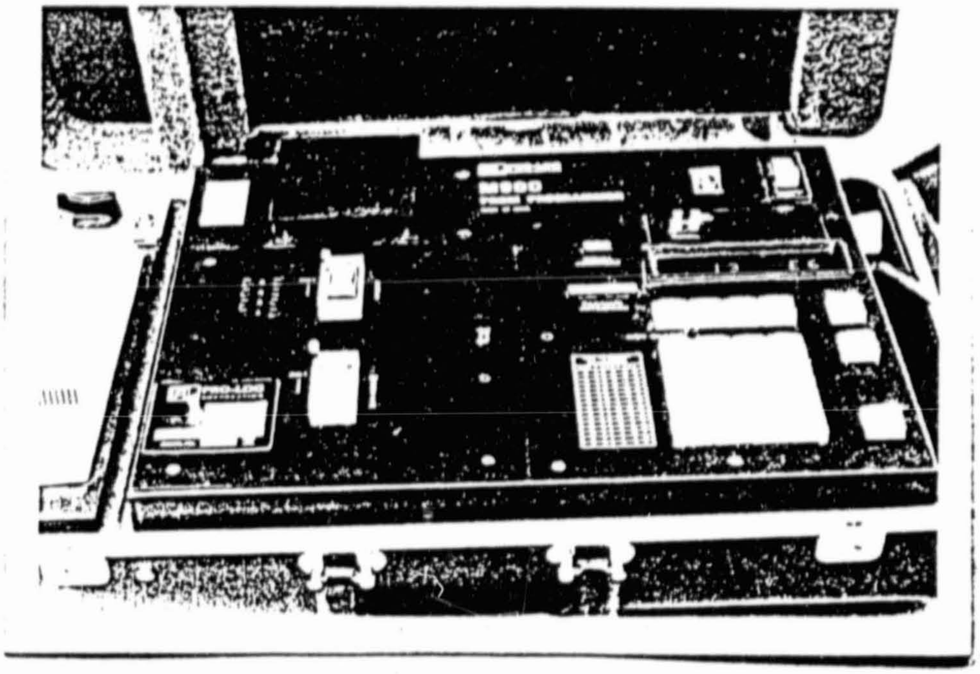


Figure 4. M900 programmer with PM9001A personality module.

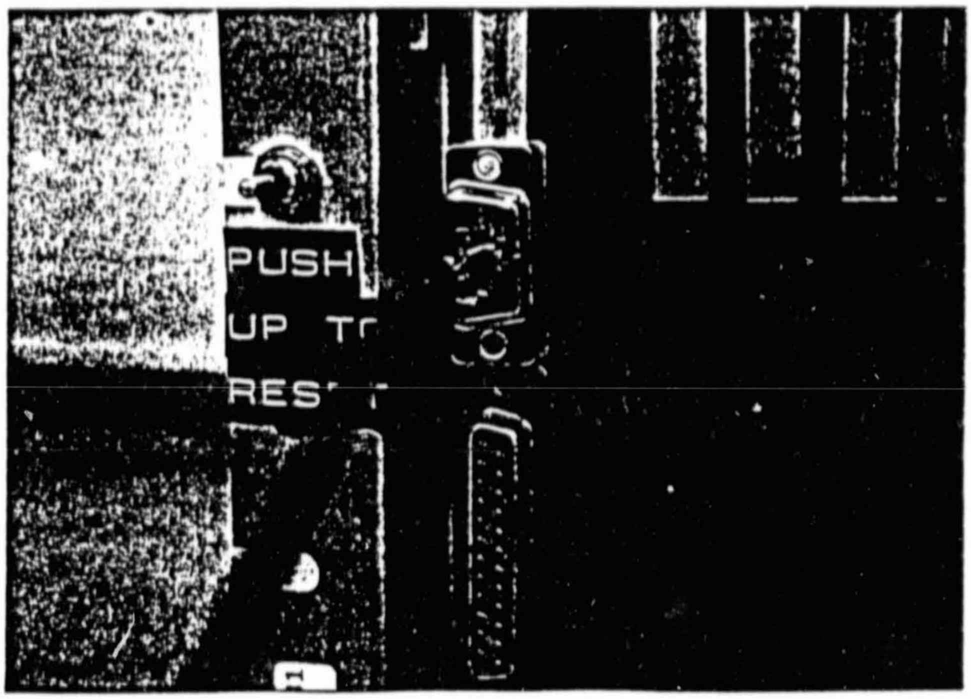


Figure 5. 20 milliamp connection between DEC-Writer II and 8407 serial interface card.

USART and programmable interval timer card.- This card was designed and wirewrapped to serve two purposes. To provide timing and serial-to-parallel conversion for interface circuit between 8407 Interface card and DEC-Writer. Also to provide a hardware timing device to interrupt Central Processor Unit (CPU) at a periodic time interval so the next moisture content reading could be taken. The schematic for this board is in Appendix A along with the interface connection between board and the microprocessor card rack system.

A 8253 Programmable Interval Timer is used to measure one minute time period between moisture readings. A decision was made to use hardware timing because timing had to be taking place while microprocessor was executing software controlling moisture monitoring system. Since there were many paths that could be taken during execution of software it would be very difficult to handle all of the various software timing combinations that were possible and obtain an accurate time interval. With the 8253, a value for one minute delay is loaded into the 8253 counter and when decremented to zero an interrupt request is sent to CPU indicating it is time to take a moisture content reading.

The 8253 has three sixteen bit counters and one eight bit control word register. Two counters addressed by ports 24 and 25 are used to count the one minute delay period. The counter addressed by Port 26 is used to control timing for the 8251 Universal Synchronous/Asynchronous Receiver/Transmitter (USART) for the baud rate at which the microprocessor and DEC-Writer communicate. Port 27 addresses the control word register which is used to program the programmable interval timer. So the programmable interval timer can be controlled by software to determine the type and length of timing delay that is produced.

Timing and conversion from serial-to-parallel data for communication between microprocessor and DEC-Writer is done by the 8251 USART. The 8251 has an eight bit status register and an eight bit data register. The status register is addressed by port 21 and is accessible by the CPU to determine if the 8251 is ready to receive or transmit data. Data is transmitted or received through the data register which is addressed by port 20. The 8251 does conversion from parallel to serial for transmission from microprocessor and from serial to parallel for transmission into microprocessor.

A picture of the wire wrapped board is shown in Figure 6. To access the board an IN or OUT instruction must be addressed to ports 20, 21, 24, 25, 26, 27, 2E, or 2F. Address decoding is done by a 74S138 Decoder/Demultiplexer. The USART and programmable interval timer are addressed as stated previously and ports 2E and 2F are for an LED used in debugging procedures to determine if execution of software is proceeding as expected.

Two 8226 four bit Parallel Bi-Directional Bus Drivers are used to transmit and receive data between the chips on the card and the microprocessor. On the microprocessor side of one 8226 the four least significant Data In and Data Out lines are connected for a total of eight lines. On the card side the bi-directional capability of the chip allows each Data In and Data Out line to be connected together so on this side of the chip there are only four lines. The four most significant Data In and Data Out lines are connected to the other 8226 in the same manner. The correct chip on the card is enabled to receive or transmit data by the address decoding done by the 74S138 Decoder/Demultiplexer.

On the 8253 programmable interval timer chip counter values are loaded into sixteen bit counters zero and one addressed by ports 24 and 25 respectively to obtain a one minute time delay. The microprocessor clock is used as the clock for counter one which is programmed

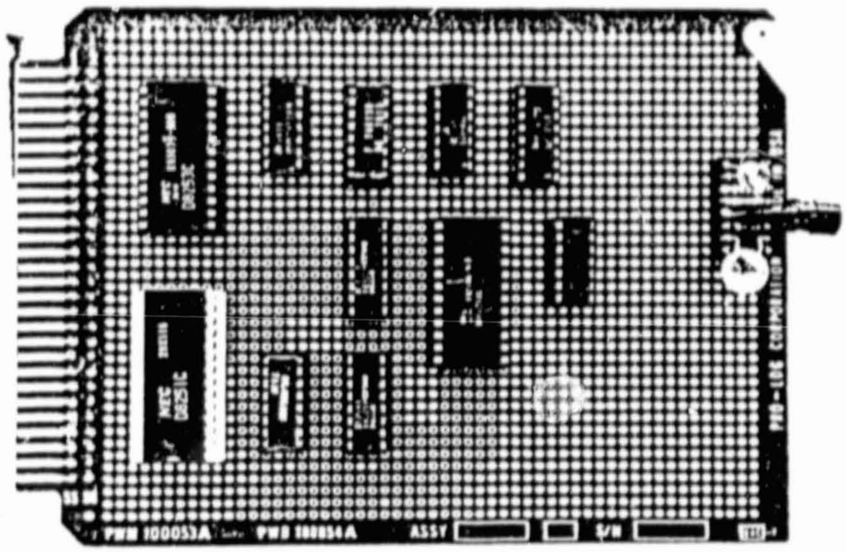


Figure 6. Wirewrapped USART and programmable timer card.

to generate a square wave form with a period of one hundredth of a second. This square wave output from counter one then acts as the clock for counter zero. When the value in counter zero is decremented to zero indicating a one minute time delay an Interrupt Request signal is sent to the CPU. The CPU then acknowledges the interrupt by sending a Interrupt signal which enables the 8212 eight bit Input/Output Port on the wire wrap card. The input lines on the Input/Output Port are connected to ground which causes a low to be transmitted on the Data Out lines. These output lines are inverted going into the CPU so the CPU sees all lines as being high which is the FF instruction for a Restart 7. When this instruction is executed the program address goes to 0038 where instructions are stored to set up one minute time delay again and take another moisture content reading.

The 8251 USART handles timing for baud rate and conversion from serial to parallel data. Timing for baud rate is provided to the USART by the output from counter two on the programmable interval timer. Parallel data from microprocessor is converted to serial data and sent on the Transmitter Data line of the 8251 to the Data Out, DO-A and DO-B, lines of the 8407 Serial Interface card to be transmitted to the DEC-Writer. Serial data from the DEC-Writer passes through the 8407 Serial Interface card on the Data In line to the Receiver Data line on the 8251 where it is converted from serial to parallel and sent to the microprocessor.

The time delay can be varied with software programming to the programmable interval timer. With some additional interface wiring between this card and the 8407 Serial Interface card communication with an RS-232 type terminal could be provided if required.

<u>Calculator Card and Interface Wiring</u> - A method of performing multi-byte addition, subtraction, multiplication, and division was needed to

do calculations required by the moisture content monitoring system. A decision was made to do this in hardware with a 90006 Calculator Card manufactured by Micro-Link Corporation to be compatible with the card rack system being used. This card operated in floating point mode and exponential mode. Also it has capabilities to do square roots, squaring, logarithm, natural logarithm, exponential, trigonometric, and inverse trigonometric functions. Command sequence is of the Reverse Polish Notation type similar to that of many hand held calculators.

Since the calculator card is compatible with the card rack system, its size, power requirements, and power input pins were such that it would fit inside the card rack. The calculator card was put in slot twelve of the card rack. Backplane wiring between the calculator card and card rack system are shown in Appendix A. Power was already bussed to the correct pins by the card rack.

The microprocessor needed to read three status signals and four data lines from the calculator card. READY status had to be checked to determine if calculator card was prepared to receive instructions from the microprocessor. DATA AVAILABLE* status had to be checked to see if calculator card had data available to be read by microprocessor. And a ERROR* status line was available to signal microprocessor an error had occurred during the calculation procedure on the calculator card. The * after DATA AVAILABLE, and ERROR indicates that these lines are active when they are low. The four data lines are for the microprocessor to read a digit of the answer from the calculator during each read procedure. These seven lines were connected to the microprocessor system through Input port 03. The status signals were connected to input lines 8, 7, and 6 with the four data lines connected to input lines 4, 3, 2, and 1.

Nine output lines were required to carry instructions from the microprocessor to the calculator card. One of these, the RESET* signal, was

connected through Output port 02 on line 8. The other eight signals were connected to Output port 03. Line eight of Output port 03 was connected to the STROBE-OUT signal which informed calculator card the data on its data lines had been read and new data was to be put on the data lines. The STROBE-IN signal informed calculator card to read the instructions on the instruction lines and it was connected to line seven of Output port 03. There were six lines required to send the six bit instruction code to the calculator card and these were connected to the remaining six lines of Output port 03.

Solenoid Selection and Interface Wiring - To operate the moisture meter used in the moisture content monitoring system a LOAD button had to be depressed to allow grain sample to drop into test cell and a UN-LOAD button had to be depressed to allow the grain sample to drop out of the test cell. These two buttons opened and closed the top and bottom of the test cell by mechanical linkages. To operate the opening and closing of the top and bottom of the test cell some type of device was needed to electrically operate the mechanical linkage to allow remote control. A decision was made to use push type solenoids to depress the LOAD and UNLOAD buttons. The travel distance of the buttons was measured to be five sixteenths of an inch. The force requirement to depress the buttons was measured with a Chatillon Model DPP push/pull gauge owned by the Agricultural Engineering Department, property control number 89539. The force required to depress the LOAD button with the 250 gram sample in the weighing hopper was two and one half pounds. The force required to depress the UNLOAD button with the grain sample in the test cell was six pounds. Gaurdian Electric manufactured two solenoids which fit these criterion. To operate the LOAD button a 16P 120 VAC Intermittent type solenoid with force characteristics of 88 oz @ 1/8 inch and 22.5 oz @ 3/4 inch was purchased. To operate the UNLOAD button a 18P 120 VAC

continuous type solenoid with force characteristics of 137 oz @ 1/8 inch and 90 oz @ 7/8 inch was purchased. These solenoids are shown mounted on the moisture meter in Figure 7.

So that the microprocessor could control the turning on and off of the solenoids a PRO-LOG 8404-4 Triac card was bought and interfaced into the system. This card was made to interface with the microprocessor card rack system and has four solid state power relays (TRIACS) mounted on it. This provides buffered TTL control of four 240 VAC two Amp isolated circuits. Each TRIAC is actually capable of switching up to ten Amps of current if mounted outside the card rack system on a heat sink to dissipate the additional heat generated by the higher currents. A Fluke Digital Multimeter with current measuring clamp was used to measure current being drawn by solenoids while they were on. The LOAD solenoid required 0.65 Amps while on and the UNLOAD solenoid required 0.37 Amps while on so there is no problem with heat dissipation. In the moisture content monitoring system the Triac card is mounted outside the card rack system in a BUD Box to eliminate the danger of high voltage coming into contact with the microprocessor system.

The LOAD solenoid is operated by Triac number one and the UNLOAD solenoid is operated by Triac number two. The solenoid 120 VAC wiring is connected to the triac by cutting one of the two wires going to the solenoid and connecting one end to terminal number one and the other end to terminal number two of the triac. Whenever the selected triac receives a low signal from the microprocessor it completes the circuit and allows current to flow to the solenoid turning it on, then a high signal to the triac from the microprocessor tells the triac to break the circuit, turning the solenoid off. A picture showing the wiring between the triac card and the solenoids is shown in Figure 8.

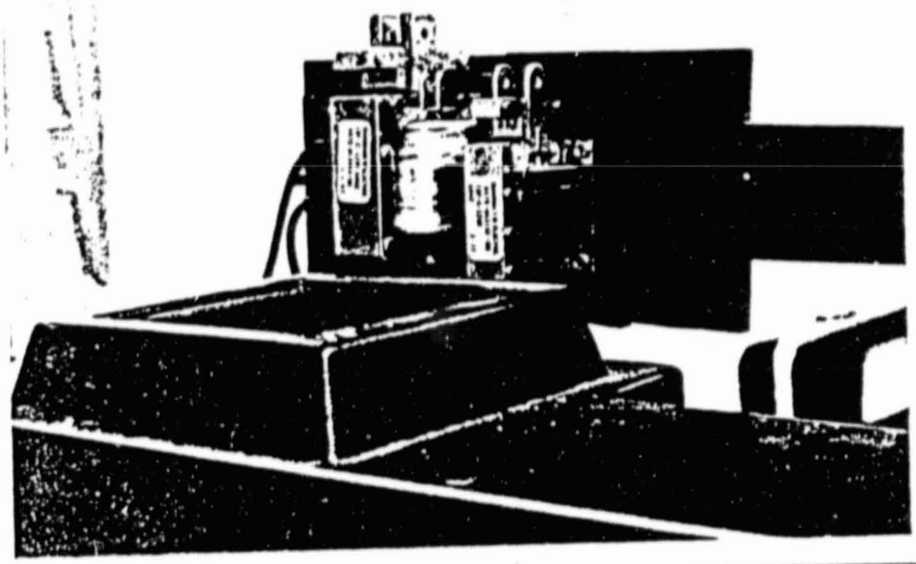


Figure 7. Solenoids mounted on moisture meter.

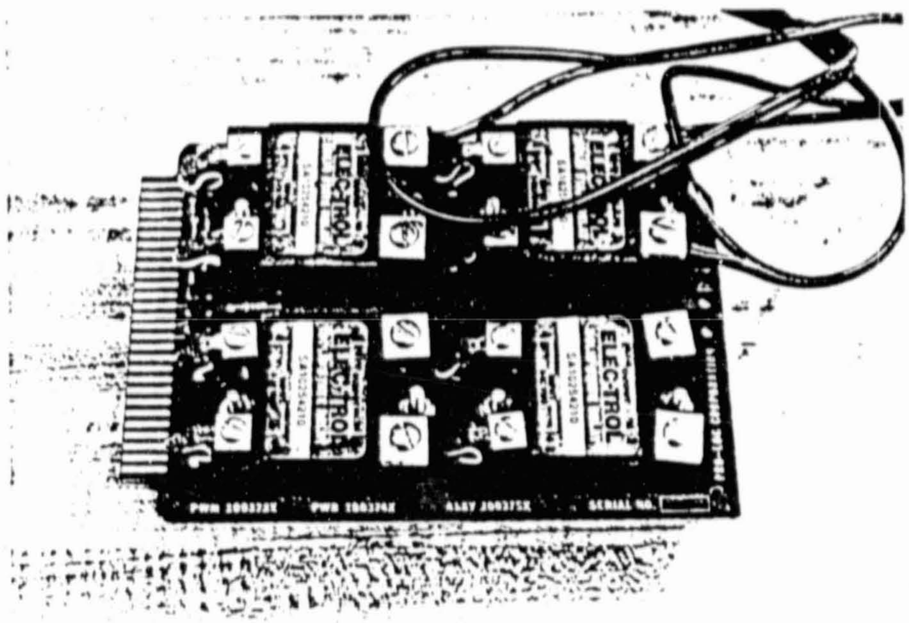


Figure 8. Electrical wiring between the 8404-4 triac card and the solenoids.

A wiring list in Appendix A shows the wiring between the microprocessor system and the triac card. Five volts from pins one and two of the card rack system is wired to pins one and two of the triac card. A ground is provided by connecting pins three and four of the triac card to pins three and four of the card rack system. Output signals from the microprocessor to the triac card are sent through Output port 01 by output lines one through four being connected to the respective triac input lines. The triacs make a circuit when a low signal is sent from the microprocessor and break the circuit when a high signal is received from the microprocessor.

Moisture Meter and Interface Wiring - A Burrows Model 700 Digital Moisture Computer was used as the moisture content sensor. This is a capacitance type moisture meter which requires a 250 gram sample and it performs automatic temperature correction for the temperature of the grain. It gives a direct digit l moisture content reading, using nixie tubes, within two to twelve seconds depending on temperature correction required. This is a wet basis moisture content reading which is the moisture content basis grain is sold on in the United States. Wet basis moisture content is represented by the equation

$$MC\ (\%) = \frac{\text{wt. of water}}{\text{wt. of water + wt. of dry material}} \times 100 \qquad (2)$$

This particular model also has a printer output provided which outputs the moisture content reading in Binary Coded Decimal form to the printer connector which is used to interface the moisture meter to the microprocessor so the microprocessor can get the moisture content reading from the moisture meter.

The data in Binary Coded Decimal form is sent to the printer connector on twelve lines for the three digit moisture content reading. Four lines carry the tenths of percent digit, four lines carry the units of

percent digit and the remaining four lines carry the tens percent digit. Each of the four lines for a digit will carry a high (1) or a low (0). A low represents zero and a high on line one represents one, a high on line two represents two, a high on line three represents four and a high on line four represents eight. An output to the printer connector of 0001 0101 1001 represents a moisture content reading of 15.9%. Pictures showing the printer connector on the back of the moisture meter and the connector with wiring to interface the moisture meter to the microprocessor are shown in Figure 9.

Appendix A contains the wiring list showing the connections between the microprocessor and the printer connector on the back of moisture meter. Wiring from the backplane of the card rack system is connected to a Cinch 57-30360 36 pin connector which attachs to the printer connector on the back of the moisture meter. The tenths of a percent digit is input through lines one through four of Input port 00. The units of a percent digit is input through lines five through eight of Input port 00. And the tens of a percent digit is input through lines one through four of Input port 01. A ground is provided by connecting pins three and four from the card rack backplane to the two ground lines of the printer connector.

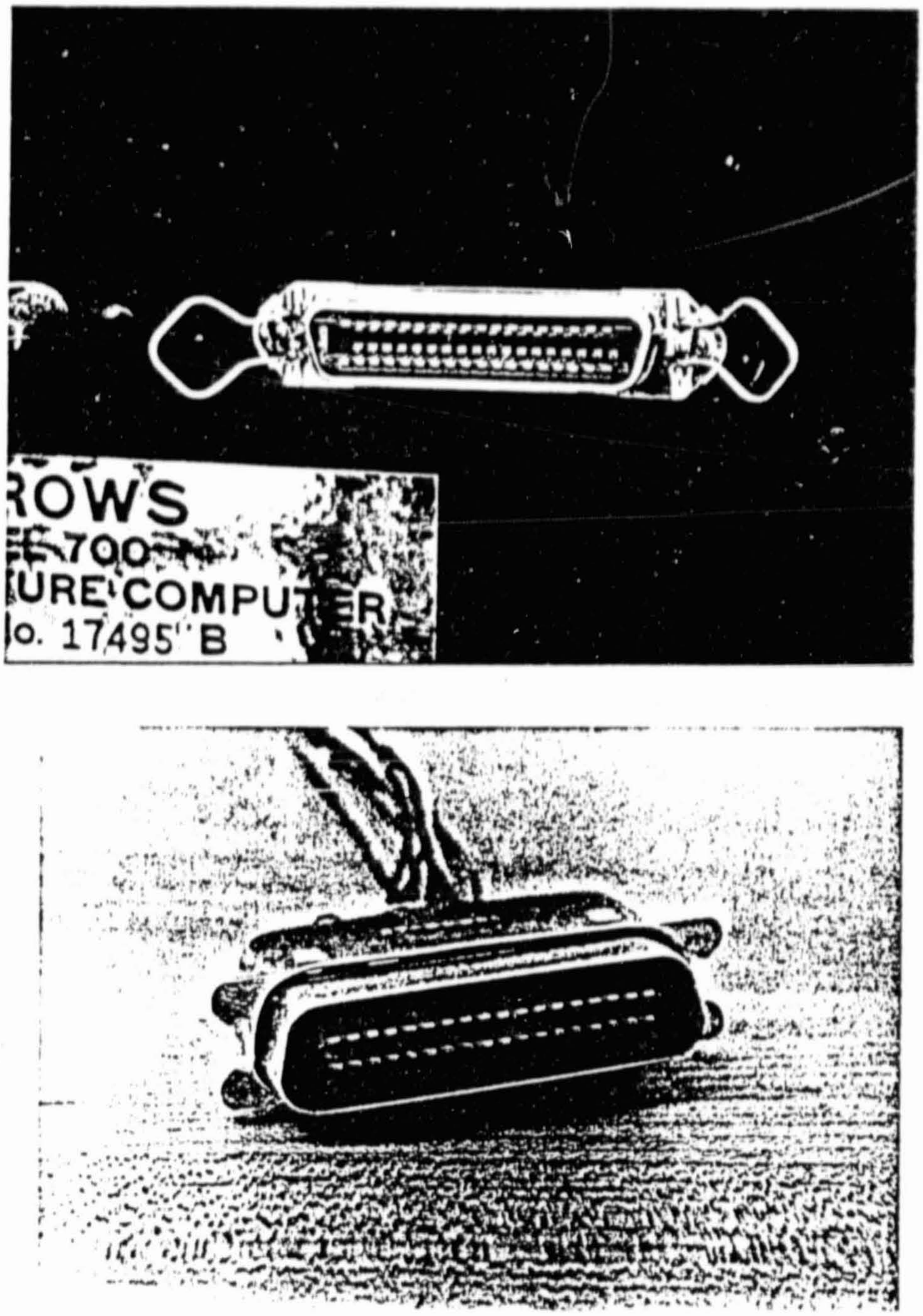


Figure 9. Printer connector on back of moisture meter and connector which interfaces moisture meter to the microprocessor.

System Software

Software Development

During this phase a monitor developed by Dr. R. L. Schafer of the National Tillage Machinery Laboratory was used. The monitor contained utility subroutines which provided control of the microprocessor system from the DEC-Writer keyboard. The monitor initializes the microprocessor stack and sets up the USART in preparation for data transmission between microprocessor and DEC-Writer. Subroutines in the monitor program then handle data transmission between microprocessor and DEC-Writer. RAM memory on the microprocessor can be programmed from the DEC-Writer keyboard or read in from a ASR-33 teletype which has a paper tape reader on it. The ASR-33 operates on a 20 milliamp circuit so it interfaces with the microprocessor through the same nine pin connector on the 8407 serial interface card used by the DEC-Writer. Also a tape showing the programming in the RAM memory can be punched using the paper tape punch on the ASR-33. The monitor has a subroutine which controls the punching or reading of the paper tape. Programming contained in memory can be printed out on paper at the DEC-Writer by the monitor. Where single character substitution is required in RAM, it is provided by the monitor. The monitor also controls where execution of the program begins. With these control subroutines provided by the monitor, programming and debugging of the software is more easily accomplished.

Software development was done by programming a subroutine into RAM, executing it, and doing the debugging required to obtain correct execution with program in RAM. When the subroutine operates correctly it is programmed into a PROM by the PROM programmer. The PROM is inserted into the correct page address on a ROM card and the next subroutine or section of program

is developed in RAM. This process continues until all the software has been developed. Doing the debugging in RAM is faster and easier because a RAM value can be changed from the DEC-Writer keyboard with the system remaining on. If a change in ROM is required the system must be turned off, ROM card pulled from card rack, old PROM removed from card, new PROM programmed and inserted onto card, put card back into card rack, and turn system back on. This can be a time consuming process where many changes in programming are required.

During the software developmental phase the 8117 RAM card was used in the card rack for storing the program. This card has 4096 bytes of eight bit RAM capacity used in increments of 1K (1024 bytes). For full capacity of the card thirty-two 2102-4 static RAM chips are required and supply addressing from 4000 to 4FFF. This provided all of the RAM memory space required during developmental phase.

Software System Overview

The system is set up to monitor two types of situations, one when the grain is being unloaded into a bin at the terminal and the other is when grain is being loaded from the terminal into a truck, boxcar, barge, or ship to be transferred to another terminal in the United States or for export marketing. When the grain is being unloaded into the terminal the system monitors individual moisture content readings at one minute time intervals, and calculates and monitors average moisture content for the grain that has been unloaded. In the loading out situation the system does the above and also at selected periodic intervals calculates a required moisture content for the remaining grain to be loaded to obtain a desired final average moisture content for the complete shipment.

This requires inputs by an operator to the system for the date, beginning time, upper and lower limits for individual moisture content reading,

upper and lower limits for average moisture content, and the grain destination. These inputs are required by both types of monitoring situations. To monitor the loading situation additional inputs are required for desired final average moisture content, approximate number of readings to be taken, and the interval between calculations of required moisture content of remaining grain to obtain the desired final average moisture content. With these initial input parameters the system is ready to start the monitoring operation.

Using the solenoids mounted on the moisture meter a grain sample is loaded into the test cell, a moisture content reading is taken, and the grain sample is unloaded out the bottom of the test cell. This system is set up without any method for pulling a grain sample. During testing of the system grain samples were provided manually. To incorporate the system into a grain terminal a device would have to be developed to pull the grain samples. After development of the sample pulling device it could be controlled by the microprocessor system. After the moisture content reading is taken the microprocessor calculates average moisture content.

Each minute a moisture content reading is taken, average moisture content is calculated, and a data record is output to the DEC-Writer. Contained in the data record is the date, time, individual moisture content that was just taken, newly calculated average moisture content, and the grain destination. The individual moisture content reading that was just taken is checked for being above the high limit for individual moisture content or below the low limit for individual moisture content reading. If the individual moisture content reading is outside either limit an appropriate message is printed by the DEC-Writer. The same check is done on the average moisture content and if it is above or below its limits the appropriate message for average moisture content is printed by the DEC-Writer.

If the loading out situation is being monitored [illegible] proper number of readings has passed, calculations are performed to determine the required moisture content for the remaining grain to obtain the desired final average

moisture content. The required moisture content for remaining grain that has been calculated is output to the Dec-Writer. If all the expected readings have been taken a message is output to the DEC-Writer stating that the predicted number of readings have been taken. After this message is sent the system continues to monitor the moisture content and average moisture content, but it doesn't calculate the required moisture content for grain anymore.

At this point in the program the moisture content reading has been taken, and all moisture content calculations required have been made. The remaining section of the program updates the time to keep it current. This is done by adding one minute to the minutes value of the time. The first half of the day is AM and the second half of the day is PM. The software keeps the correct day of the month, hour of the morning or afternoon, and minutes. If a monitoring operation is started on the last day of the month and finished the next day, the monitor will list the same month with the day incremented. An example is starting on 01-31-80 and finishing the next day. Instead of showing 02-01-80 the records will contain 01-32-80. But this shouldn't be a handicap for keeping records if this feature is understood.

Main Program - The monitor that was used during software development isn't in the moisture content monitoring system. All of the control functions that it provided that are required by the moisture content monitoring system were adopted into the system software. The initialization of the stack and setting up the USART for data transmission which were formerly done by the monitor is done by the software at the beginning of the main program. The main program software which was developed and is being discussed in this section is in Appendix B. This is the programming stored in memory space 0000 through 003F. Random access Memory (RAM) is provided

by the 1024 bytes of RAM contained on the 8811A Processor card thereby eliminating the need for a RAM card in the system. The 1K of RAM that is on the Processor card is in addresses 3000 through 33FF.

The first instruction in the main program initializes the stack pointer to 33CD. This allows this memory space and any below it to be used as a sixteen bit stack to store subroutine return addresses and any other addresses or data required by the system software. Whenever a jump to subroutine instruction is executed the return address which is in the program counter is automatically pushed onto the stack. Any addresses that were already there are pushed down on the stack to a lower stack address. When the Return instruction is executed the top address on the stack is pulled off and put into the program counter and this address is where program execution continues. Any addresses that were stored on the stack below the subroutine return address are moved upwards toward the top of the stack when the subroutine return address is pulled off the top of the stack. Data or addresses stored in any of the register pairs can be pushed onto or pulled off of the stack in the same manner. More information on how the stack pointer works can be found in The Designer's Guide to Programmed Logic.

After the stack pointer has been initialized instructions must be sent to the USART preparing it to start data transmission. Timing must be provided for the USART so that it can communicate with the DEC-Writer at a 110 baud rate. A control word of B6 is sent to the programmable interval timer since Counter two on it is to be used as the clock for the USART. The B6 command instructs programmable interval timer to select Counter two, to Read or Load Least Significant Byte first, then Most Significant Byte, and operate as a square wave rate generator with a sixteen bit binary counter. Counter two is then loaded with a count value of 022B which will provide a square wave which when multiplied by the sixteen times baud rate factor provides a clock for the USART to transmit at a 110 baud rate.

With the timing provided, the USART instructions are sent to the USART to set up its mode of operation. The USART is set up for Asynchronous transmission format. The control word sent to USART to set up mode is CE which instructs the USART that data should be transmitted with two stop bits, eight data bits, parity disabled, and a baud rate factor of sixteen times the clock being used. Immediately after the USART mode has been programmed a command instruction controls the actual operation of the format selected by mode instruction. The command instruction is 37 which forces REQUEST TO SEND* output to zero, resets error flags, forces DATA TERMINAL READY* output to zero, and enables receive enable and transmit enable. After receiving the command instruction the USART is ready for data transmission. Additional information on the programmable interval timer chip and the USART chip and their programming can be found in the Intel MCS-80 User's Manual.

Whenever a Restart 7 instruction is encountered the microprocessor automatically executes the instruction stored at address 0038. The Restart 7 instruction is put on the data lines when the interrupt occurs signaling the end of a one minute delay. The instruction stored at address 0038 is a Jump instruction to a subroutine which resets the interrupt to occur after one minute delay. After resetting the interrupt this routine directs the microprocessor system to the correct address to continue the moisture content monitoring process. More information will be given concerning the Interrupt Reset routine in the section covering subroutines.

The software that has been discussed is software that replaced functions previously supplied by the monitor. After executing these instructions the microprocessor jumps to the main program that is stored in memory space through OBFF. The software in the main program will now be discussed in the order it would be executed by the microporcessor.

Memory space that will be used for storing data record, input parameters,

and calculation space is copied from ROM memory 1600-170F into RAM memory 3000-310F. This material is moved into RAM so that it can be written to the microprocessor. Addressess 3000-304F has 0D and 0A programmed in the first two addresses which are ASCII instructions for DEC-Writer carriage return and line feed. The remaining addresses are blank spaces (20) and will be programmed with the data, time, individual moisture content reading, average moisture content, and grain destination when these parameters are obtained. Memory space 3050-310F is originally programmed with Nulls (00) unless otherwise specified as shown in Table 1. This memory space will be programmed as the microprocessor receives the values that should be stored in each memory address. Table 2 contains the addresses and what will be programmed in each space. There are several different forms in which data are stored. These are ASCII, Actual, Calculator, and Answer from calculator form. A value of 15.9 would be represented in ASCII as 31 35 2E 39, in Actual form as 01 05 09, in Calculator form as 01 05 0A 09, and in Answer from calculator form as 00 0A 01 05 09. More will be discussed about these forms and why they are used as they are encountered during the discussion of the main program.

Now the microprocessor is ready to accept the input parameters which will be used during operation of the moisture content monitoring system. This is done by the microprocessor outputting a question to the DEC-Writer concerning the particular input parameter of interest. Hexadecimal coding in ASCII form is stored in memory space 1000-156D for each question or statement that the microprocessor uses for communication. Each letter, number, or symbol requires one byte of hexadecimal coding in ASCII form for identification. Figure 10 shows table from The Designer's Guide to Programmed Logic which was used to find hexadecimal coding for letters, numbers, symbols and control characters. The questions and statements used for communication by the microprocessor are shown in Table 3 with the memory

TABLE 1

INITIAL DATA STORED IN OUTPUT RECORD AND CALCULATION MEMORY SPACE

ROM ADDRESS	DATA Instruction	DATA Hexadecimal coding
1600	CR	0D
1601	LF	0A
1602-164E	Blank	20
164F-1669	Null	00
166A	Enter	21
166B-1673	Null	00
1674	Addition	39
1675-1683	Null	00
1684	Enter	21
1685-1687	1.0	01,0A,00
1688	Addition	39
1689-1697	Null	00
1698	Enter	21
1699-169D	Null	00
169E	Division	3C
169F-16B1	Null	00
16B2	Enter	21
16B3-16B6	Null	00
16B7	Multiply	3B
16B8-16C6	Null	00
16C7	Enter	21
16C8-16CC	Null	00
16CD	Subtraction	3A
16CE-16DC	Null	00
16DD	Enter	21
16DE-16E6	Null	00
16E7	Subtraction	3A
16E8-16EC	Null	00
16ED	Division	3C
16EE-16F9	Null	00
16FA-16FC	500	05,00,00
16FD-16FF	050	oo,05,00
1700-170B	Null	00
170C	LF	0A
170D	Null	00

TABLE 2

MEMORY SPACE ALLOCATED FOR DATA VARIABLES AND CONSTANTS
TO BE USED IN OUTPUT RECORD AND FOR CALCULATIONS

RAM ADDRESS	DATA	Form
3000-3001	CR, LF	ASCII
3002-3009	Date	ASCII
3001-3014	Time	ASCII
301C-301F	Individual MC reading	ASCII
302B-302E	Average MC value	ASCII
3036-204E	Grain destination	ASCII
304F	Null	00
3050-3052	Upper limit for individual MC reading	Actual
3053-3055	Lower limit for individual MC reading	Actual
3056-3058	Individual MC reading	Actual
3059-305B	Upper limit for average MC	Actual
305C-305E	Lower limit for average MC	Actual
305F-3061	Average MC value	Actual
3062-3065	Average MC value	Calculator
3066-3069	Individual MC reading	Calculator
306A	Enter	Calculator
306B-3073	Previous total of MC reading	Calculator
3074	Addition	Calculator
3075-307E	Present total of MC readings	Answer
307F-3083	Previous number of readings	Calculator
3084	Enter	Calculator
3085-3087	1.0	Calculator
3088	Addition	Calculator
3089-308E	Present number of readings	Answer
308F-3097	Present total of MC readings	Calculator
3098	Enter	Calculator
3099-309D	Present number of readings taken	Calculator
309E	Division	Calculator
309F-30A3	Average MC value	Answer
30A4-30A8	Number of predicted readings	AASCII
30A9-30AC	Desired final average MC	ASCII
30A1-30B1	Number of predicted readings	Calculator
30B2	Enter	Calculator
30B3-30B6	Desired final average MC	Calculator
30B7	Mulitiplication	Calculator
30B8-30C1	Desired total of MC readings	Answer
30C2-30C6	Number of predicted readings	Calculator
30C7	Enter	Calculator
30C8-30CC	Present number of readings taken	Calculator
30C1	Subtraction	Calculator
30CE-30D3	Number of readings remaining	Answer
30D4-30DC	Desired total of MC readings	Calculator
30DD	Enter	Calculator
30DE-30E6	Present total of MC readings	Calculator
30E7	Subtraction	Calculator
30E8-30EC	Number of readings remaining	Calculator
30ED	Division	Calculator
30EE-30F2	Desired average MC for remaining grain	Answer
30F3-30F6	Desired average MC for remaining grain	Calculator

TABLE 2
(continued)

30F7	Number of readings interval	Decimal
30F8	Number of readings interval	Hexadecimal
30F9	Number of readings interval counter	Hexadecimal
30FA-30FC	High limit for a valid reading (05,00,00)	Actual
30FD-30FF	Low limit for a valid reading (00,05,00)	Actual
3100-3102	Desired final average MC	Actual
3103-3107	Individual MC reading	ASCII
3108-310C	Desired final average MC	ASCII

HEX				MSD p = 1	8	9	A	B	C	D	E	F
				p = 0	0	1	2	3	4	5	6	7
	BITS			b8	p	p	p	p	p	p	p	p
				b7	0	0	0	0	1	1	1	1
				b6	0	0	1	1	0	0	1	1
LSD	b4	b3	b2	b1 \ b5	0	1	0	1	0	1	0	1
0	0	0	0	0	NUL	DLE	SP	0	@	P	`	p
1	0	0	0	1	SOH	DC1	!	1	A	Q	a	q
2	0	0	1	0	STX	DC2	"	2	B	R	b	r
3	0	0	1	1	ETX	DC3	#	3	C	S	c	s
4	0	1	0	0	EOT	DC4	$	4	D	T	d	t
5	0	1	0	1	ENQ	NAK	%	5	E	U	e	u
6	0	1	1	0	ACK	SYN	&	6	F	V	f	v
7	0	1	1	1	BEL	ETB	'	7	G	W	g	w
8	1	0	0	0	BS	CAN	(	8	H	X	h	x
9	1	0	0	1	HT	EM	)	9	I	Y	i	y
A	1	0	1	0	LF	SUB	*	:	J	Z	j	z
B	1	0	1	1	VT	ESC	+	;	K	[	k	{
C	1	1	0	0	FF	FS	,	<	L	\	l	|
D	1	1	0	1	CR	GS	-	=	M	]	m	}
E	1	1	1	0	SO	RS	.	>	N	^	n	~
F	1	1	1	1	SI	US	/	?	O	_	o	DEL

CONTROL CHARACTERS	
NUL	Null
SOH	Start of Heading
STX	Start of Text
ETX	End of Text
EOT	End of Transmission
ENQ	Enquiry
ACK	Acknowledge
BEL	Bell (audible or attention signal)
BS	Backspace
HT	Horizontal Tabulation (punched card skip)
LF	Line Feed
VT	Vertical Tabulation
FF	Form Feed
CR	Carriage Return
SO	Shift Out
SI	Shift In
DLE	Data Link Escape
DC1	Device Control 1
DC2	Device Control 2
DC3	Device Control 3
DC4	Device Control 4 (Stop)
NAK	Negative Acknowledge
SYN	Synchronous Idle
ETB	End of Transmission Block
CAN	Cancel
EM	End of Medium
SUB	Substitute
ESC	Escape
FS	File Separator
GS	Group Separator
RS	Record Separator
US	Unit Separator
DEL	Delete

Figure 10. ASCII code assignments and TTY character set.

TABLE 3

MEMORY SPACE ALLOCATED FOR THE ASCII CODING OF THE QUESTIONS AND MESSAGES TO BE PRINTED

ROM Address	Question or Statement
1000-1015	What is the data?
1016-103C	What time will the readings begin?
103D-1084	What is the upper limit for an individual moisture content reading?
1085-10CC	What is the lower limit for an individual moisture content reading?
10CD-1106	What is the upper limit for average moisture content?
1107-1140	What is the lower limit for average moisture content?
1141-115D	Is the grain being unloaded?
115E-117F	What bin is grain going into?
1180-11A4	What is the grain being loaded onto?
11A5-11E8	Approximately how many readings will it take to load the grain?
11E9-1220	What is the desired final average moisture content?
1221-1252	Individual moisture content reading is too high.
1253-1283	Individual moisture content reading is too low.
1284-12AA	Average moisture content is too high.
12AB-12DD	Average moisture content is too low.
12D2-131C	Title for output record.
131D-1394	How many readings should pass between each calculation of desired moisture content for the remainder of the grain?
13C2-1405	The remainder of the grain should have an average MC precent of ø.
140D-1443	The predicted number of readings have been taken.
1445-146B	Excessively high reading, Ignored øø.
146D-1492	Excessively low reading, Ignored øø.
1494-14CA	Desired final avg MC is outside limits for avy MC øø.
14CC-1517	Low indiv MC reading limit is greater than high indiv MC reading limit.
1519-1550	Low avy MC limit is greater than high avg MC limit.
1552-155F	Wait mode
1560-156C	Run mode

addresses where they are stored. A question is output by the microprocessor by the BUFOUT subroutine which operates by starting at a specified address and outputting to the DEC-Writer whatever data it sees at this address and following addresses until a NOP (00) is encountered signaling the end of data transmission. A NOP is stored in memory at the end of each question or statement. With this method of outputting questions the microprocessor can prompt the operator concerning the input parameter that is desired.

The first question asked by the microprocessor is to obtain the date. The date is typed in on the DEC-Writer by the operator in the form of a two digit month value, dash, two digit day of the month value, dash, and a two digit year value. For example the eighth day of February of 1980 would be typed in as 02-08-80 with a carriage return to indicate that this is all of the data to be input. This data is stored in ASCII form in memory space 3002-3009 as part of the data record. The data is left in ASCII form because this is the type of coding the DEC-Writer uses to transmit and receive data.

The next input asked for by the microprocessor is the starting time for the monitoring operation. It's input format is two digits for hour of day, colon, two digits for minutes, space and a AM or PM designation for midnight to noon or noon to midnight. After every input that the microprocessor asks for is typed in there should be a carriage return to signal end of operator reply. If the starting time is three o-clock in the afternoon the input format should be 03:00 PM. This data is stored in memory space 300D-3014 as part of the data record and is also left in ASCII form.

A question is asked concerning the upper limit for individual moisture content reading. This parameter will be used to check the individual moisture content reading for being above a certain desired value. This will give an indication of the variation of the moisture content of the grain and if a significant amount of it is wet enough to cause spoilage. The input format for upper limit for individual moisture content reading is

two digits, a decimal, and a tenths digit. An example is 9.6 would be input as 09.6. This value is stored in the memory space indicated in Table 2 in Actual form. Actual form is used because it is easier to compare high or low limits with the actual moisture content reading in this form. The next input required is low limit for individual moisture content reading. It is input and stored in the same format as used for upper limit for individual moisture content. A check is done at this point to determine if the input for low limit for individual moisture content reading is actually less than upper limit for individual moisture content reading. This is done to help eliminate some operator error in typing in a wrong digit. If lower limit for individual moisture content reading isn't less than upper limit for individual moisture content reading then a message is output at the DEC-Writer stating that this has occurred and program execution jumps back to repeat questions concerning upper and lower limits for individual moisture content reading and to obtain the correct inputs. If low limit for individual moisture content reading is less than upper limit for individual moisture content reading program execution continues in a sequential manner.

Next, inputs for upper and lower limit for average moisture content are requested, typed in, and stored in the same manner as upper and lower limit for individual moisture content reading. The high and low limit for average moisture content provides a check on the variance of average moisture content and also an indication of any extreme upward or downward trend in the average moisture content. After upper and lower limit for average moisture content has been input a check is done to determine if lower limit for average moisture content is less than upper limit for average moisture content. This check proceeds in the same manner as the one done on upper and lower limit for individual moisture content reading.

A 52 which is an ASCII R is pushed onto the stack to indicate system is in RUN mode. This value is checked during the INTR-RESET routine to determine if system is monitoring moisture content or in WAIT mode and simply doing time keeping. The WAIT mode for just doing time keeping is provided in case something happens to stop grain flow. This could happen if machinery is being switched to bring grain from a different bin to change the moisture content of grain being loaded out or if a breakdown occurs inside the grain terminal.

All of the previous input parameters are required whenever the moisture content monitoring system is monitoring either a loading or unloading situation. At this point the monitoring system needs to know what type of situation is being monitored because different inputs are required for whichever type of situation is being monitored. So a question is output to the DEC-Writer asking if the grain is being unloaded. The first letter of the reply from the operator is saved on the stack to be checked for Y or N indicating yes or no. The remainder of the operator's reply is read in and printed at the DEC-Writer until a carriage return signal is found signaling the end of the reply. The first letter of the reply which was stored on the stack is pulled off and checked for being Y or N. If the letter is N for No the program address jumps to the section of the program to determine the remaining input parameters required for a loading out situation. If the letter is Y for Yes the program address jumps to the section of program to obtain the remaining input parameters required for a unloading situation. If the letter is neither Y or N the operator didn't give the correct response so the program address jumps to repeat the section of program which determines if grain is being unloaded.

If the unload situation is being monitored a question asking what bin the grain is going into is output to the DEC-Writer. The operator replies by typing in a descriptive number or name for where the grain will be stored

in the grain terminal. Twenty five spaces in memory provide ample space to store the descriptive number or name showing where grain is stored. Where the grain is being put in the grain terminal is included on the output record so it can be used for later reference to find the moisture content of grain in individual bins in the grain terminal. Program address then bypasses the programming concerning the loading out situation and continues executing the monitoring system program.

If a loading out situation is being monitored some control over the final average moisture content of the grain shipment being loaded out is desired. To do this, at a periodic interval calculations are performed to determine what the required moisture content of the remaining grain to be loaded has to be in order to obtain a desired final average moisture content for the grain shipment. The equation for this calculation is

$$\text{Required MC for remaining grain} = \frac{\text{Desired total of MC readings} - \text{Present total of MC readings}}{\text{number of readings remaining}} \qquad (3)$$

This equation required additional input parameters for desired final average moisture content, approximate number of readings that will be taken during loading of grain, and the reading interval at which the calculation will be made. Also a question is asked concerning what the grain shipment is being loaded onto.

Using the same procedure as used for previous input parameters the microprocessor outputs questions to the DEC-Writer and brings in and stores data typed in on the keyboard of the DEC-Writer for the following input parameters: what the grain is being loaded onto, how many readings will be taken during loading of the grain, what is the desired final average moisture content of the grain shipment, and the number of readings interval between each calculation of required moisture content for remaining grain.

Data telling what the grain shipment is being loaded onto is stored in ASCII and used as part of the output record. Twenty five spaces of memory

storage are provided so this data can contain up to twenty five letters, numbers, or symbols. The approximate number of readings to be taken during the loading operation is stored in ASCII and converted and stored in calculator form to be used later in performing calculations. The number of readings to be taken can only be five digits in length and must include a decimal. This allows up to 9999 readings to be taken during a monitoring operation. The desired final average moisture content is input in ASCII form with two digits, a decimal, and a tenths digit. This value is stored in ASCII form and then converted and stored in calculator form to be used later in performing calculations. A check is made to determine if the desired final average moisture content is within the upper and lower limits previously input for average moisture content. If the desired final average moisture content isn't within these limits as it should be, a message stating that the desired final average moisture content is outside the upper and lower limits for average moisture content is output to the DECWriter and the question asking for the desired final average moisture content is repeated. The value for number of readings interval between each calculation of required moisture content is input as a one or two digit number without a decimal,so up to 99 readings are allowed between each calculation of required moisture content for remaining grain. The value that is input is in decimal form,but the microprocessor operates in hexadecimal so the value is converted into an equivalent hexadecimal value that requires one memory space. This counter value is stored in two different memory locations, one to serve as a counter to be decremented to determine when calculation of required moisture content for remaining grain should occur, the other to reload the counter after it has been decremented to zero.

A calculation is made to find the desired total of individual moisture contents by multiplying the desired final average moisture content by the number of readings to be taken during the grain loading operation. The calculator instruction set is shown in Appendix C. The values to be used

in making a calculation are stored in specified spaces in memory with the correct calculator operators such as enter, addition, multiplication, subtraction, and division already programmed into correct memory space. The procedure for making this calculation and following calculations is the same. A RESET signal is sent to the calculator to clear memory buffers and registers in preparation for doing the next calculation. Two NOPs are sent after the RESET signal along with a MCLR instruction which clears all registers, sets mantissa digit count to eight automatically, and instructs calculator to input and output values in floating point mode. Next, a data block containing number of predicted readings, enter instruction, desired final average moisture content, and multiply instruction is sent to the calculator. All data sent to the calculator must be in calculator form. The data is sent to the calculator by a subroutine which requires the starting memory address of data in register pair BC and a counter in register D for number of bytes of data to be sent to the calculator. A subroutine then instructs the calculator to put the answer it has calculated into an output buffer so it may be read by the microprocessor. The READ-CAL subroutine reads the answer in answer form from the calculator. A counter in register B controls the number of bytes of the answer that are read from the calculator and stored in memory beginning at the address contained in register pair DE. This data in answer form is then converted into calculator form and stored in appropriate memory space to be used in making other calculations.

At this point the microprocessor starts executing monitoring system software which is used in both grain loading and unloading situations. The title heading for the output record is printed at the DEC-Writer.

A one minute delay is programmed using counters zero and one of the 8253 programmable interval timer. A control word of 30 is sent to the control word register which instructs the programmable interval timer to select counter zero, Read/Load Least Significant Byte first, then Most Sig-

nificant Byte, interrupt on terminal count, and counter zero will be a sixteen bit binary counter. Counter zero is then loaded with a value of 1770 to be downcounted. Then a control word of 76 is sent to the control word register which instructs the programmable interval timer to select counter one, Read/Load Least Significant Byte first, then Most Significant Byte, produce a square wave rate generator, and counter one will be a sixteen bit binary counter. Counter one is loaded with a value of 2710. The clock for counter one is the one microsecond clock used by the system. With a counter value of 2710, counter one produces a square wave form with a period of ten thousand microseconds or one hundredth of a second which acts as the clock for counter zero. So every one hundredth of a second counter zero is downcounted by one. The binary value of 1770 which is in counter zero is a decimal value of six thousand. Six thousand multiplied times one hundredth of a second is sixty seconds or one minute. After the counters are set to the correct values the interrupt capability is enabled. The remainder of the system software will be executing while the one minute delay is being counted by the programmable interval timer. The remainder of the system software will be through executing and the microprocessor will be in a cycle, doing nothing, while awaiting the interrupt signaling the end of the one minute delay. When counter zero is downcounted to zero an interrupt request (INTR REQ*) signal will be sent to the CPU, signaling the end of the one minute delay, and program execution will be sent to an interrupt handler.

While the one minute delay is being counted program execution continues. A low signal is output on the Least Significant Bit of Output port 01, signaling Triac #1 to complete circuit allowing current to flow to the LOAD solenoid. This activates the LOAD solenoid so that the LOAD button is depressed allowing grain sample to fall into test cell of the moisture meter. The LOAD solenoid remains on for one second by using a software delay rou-

tine. Registers B, C, and D are loaded with appropriate values and decremented in a loop. When register B is finally decremented to zero the one second delay has been completed. A more detailed explanation of this delay routine is given in The Designer's Guide to Programmed Logic. When the one second delay has been completed a high signal is output on the Least Significant Bit of Output port 01, signaling Triac # 1 to break circuit, to turn off the LOAD solenoid, releasing the LOAD button.

The same type of delay routine as was just used is used to provide a fifteen second delay to allow moisture meter enough time to take a moisture content reading of the grain sample in the test cell of the moisture meter. After the fifteen second delay has been completed the moisture content reading taken by the moisture meter is read by the microprocessor and stored in actual form as a three digit value. A check is made to determine if the moisture content reading taken is valid. If the moisture content reading is less than 05.0 or greater than 50.0 it is considered an invalid reading. If the reading is invalid a message stating that the reading is invalid is output to the DEC-Writer along with the invalid reading and the reading is disregarded. The grain sample is dumped out of the test cell, time keeping is performed, and the system awaits the one minute interrupt signal.

If the moisture content reading is valid the ACT-ASCII subroutine is called to convert the moisture content reading from actual form into ASCII form. The moisture content reading is stored in ASCII form as part of the output record. The ACT-CAL subroutine is used to convert and store moisture content reading in calculator form to be used in calculating average moisture content.

Calculations are performed to find the present average moisture content value. This will require three calculations to be made by the calcu-

lator card. These calculations are present total of moisture content readings, number of moisture content readings taken, and the average moisture content value. These calculations are performed in the same manner as the calculation that was previously done. A listing of what is stored in calculation space is shown in Table 2 which is in an earlier part of this section.

The first calculation is to determine the total of moisture content readings. A RESET signal is sent to the calculator card and instructions for mode and mantissa digit count are then output to calculator card. Next the LOAD-W-COUNT subroutine is used to send data starting at 3066 and the following OF bytes to the calculator. This data includes individual moisture content reading, enter operator, total of moisture content values, and addition operator. OUT subroutine informs calculator to prepare to output the answer so it can be read by the microprocessor. The READ-CAL subroutine reads the answer from the calculator and stores it in answer form in the storage and calculation memory space. The answer is then converted into calculator form and stored in two different memory locations to be used for later calculations.

Next the number of readings taken is calculated using the same calculation procedure. The data that is sent to the calculator to be used in making the calculation is number of readings taken, enter operator, 1.0, and the addition operator. This calculation adds one to the previous number of readings taken to obtain the present number of readings taken. The answer is read from the calculator, stored in memory in answer form, converted to calculator form and stored in three different memory locations to be used for making more calculations.

Now the microprocessor has values required to calculate a present average moisture content. Data sent to the calculator card is the values which have just been calculated and stored in memory in calculator form The sequence of data transmission is total of moisture content values,

enter instruction, number of readings taken, and the division instruction. The answer is read and stored in answer form into memory. A special subroutine, AVG-ANS-CAL, is used to convert data from answer form into calculator form and store it into memory. The subroutine rounds the data to three digits in the form of two digits, a decimal point, and a tenths digit. An average moisture content value calculated to be 9.4556000 would be 00 0B 09 04 05 05 06 00 00 00 in answer form and would be converted to 00 09 0A 05 in calculator form or printed as 09.5 on the output record. This prevents the average moisture content value from being printed with a hundredths digit which would imply accuracy which doesn't exist. The average moisture content value is converted from calculator form into ASCII form and stored in memory as part of the output record.

The output record is printed at the DEC-Writer by the use of the BUFOUT subroutine. The output record contains date, time, individual moisture content reading, average moisture content value, and the destination where grain will be stored.

The next sequence of instructions checks the individual moisture content reading and average moisture content value for being outside the limits input at the beginning of the monitoring operation. In preparation for performing these checks the average moisture content value is converted from calculator form into actual form and stored into memory. The subroutines used to perform these checks are HI-INDIV-TEST, LO-INDIV-TEST, HI-AVG-TEST, and LO-AVG-TEST. The starting address for limit value is loaded into HL register pair and the starting address for moisture content reading or average moisture content is loaded into the DE register pair. A counter value for number of bytes to be compared is loaded into register B and when the appropriate subroutine is called it does the comparison. All values compared are in actual form to make the comparison easier and simpler. If the value being checked is outside the limits set for it, an

appropriate message stating value is outside limit is printed at the DEC-Writer and program execution continues. Otherwise program execution continues without anything being printed at the DEC-Writer.

After performing these checks for values being within their limits the system checks to determine if it needs to do a calculation for required moisture content for remainder of grain to be loaded. The direction indicator is pulled off the stack to determine if system is in loading or unloading mode. If the system is in the unloading mode the calculation isn't required so program execution jumps to the section of programming which controls the unloading of the grain sample from the test cell. If the system is in the loading mode the counter for readings interval between calculation is decremented and checked for being zero. If the counter is nonzero it isn't time to do the calculation so program execution jumps to the section of programming that controls unloading of the grain sample from the test cell of the moisture meter. If the counter is zero it is time to do the calculation to find the required moisture content of the remaining grain to be able to obtain the desired final average moisture content.

To find the required moisture content for remaining grain Equation 3 is used. A calculation must be done to find the number of readings remaining. Data that is sent to the calculator to do that calculation is number of predicted readings, enter instruction, number of readings taken, and subtraction instruction. The value for number of readings remaining is read from calculator and stored in answer form. Then it is converted into calculator form and stored into memory to be used in calculation for required moisture content of remaining grain. The CHECK-READING subroutine is used here to check if number of readings remaining is zero signaling that the predicted readings have been taken and that no more calculations for required moisture content will be made.. If the number of readings remaining is zero a message stating this is printed at the DEC-Writer and the

direction indicator on the stack is changed to indicate an unloading mode. No more calculations for required moisture content for remaining grain will then be made. All of this is done by the CHECK-READING subroutine and program execution will be sent to the UNLOAD-GRAIN section of the programming to unload grain from test cell.

If all of the predicted readings haven't been taken the calculation is done to find the required moisture content for remaining grain. The data that is sent to the calculator is total of desired moisture content readings, enter instruction, present total of moisture content readings taken, subtraction instruction, number of readings remaining, and the division instruction. This is Equation 3 put into the calculator's language. The required moisture content of remaining grain is read from calculator and stored into memory in answer form. The AVG-ANS-CAL subroutine converts this data from answer form into calculator form, rounds the required moisture content to three digits in the form of two digits, a decimal point, and a tenths digit. This is the standard format for all moisture content values used in the monitoring system. Then this required moisture content value is converted into ASCII form and stored into memory. This value is then printed at the DEC-Writer along with a statement saying that this is the required moisture content for the remainder of the grain to be able to obtain the desired final average moisture content for the grain shipment. The counter which was checked to determine when to make calculation is then reinitialized from the other counter in memory space which contains number of readings interval between doing each calculation for required moisture content of remaining grain.

Next the grain is removed from the test cell of the moisture meter. This is done by sending a low signal on the next to Least Significant Bit on Output Port 01 which signals the Triac # 2 to activate the UNLOAD solenoid to depress the UNLOAD button. A one second delay is then entered to

leave the UNLOAD button depressed for one second to allow the grain sample enough time to fall out the bottom of the test cell of the moisture meter. Then the line that controls Triac # 2 is set high and output through Output Port 01 to signal Triac #2 to turn off the solenoid. Whenever turning off either of the solenoids the two Least Significant Bits being output to Output Port 01 must be high to prevent inadvertently turning on one of the solenoids.

The remainder of the main program deals with timekeeping to maintain correct time and day of the month. The hours, minutes, and AM or PM designation or pulled out of memory in ASCII form and put into register pairs BC, DE, and HL respectively. The minutes value in register pair DE is converted from two bytes of ASCII coding into one byte which contains two decimal digits and this byte is stored in register E. An example is ASCII 33 35 which is converted into 35. The hours value is converted in the same manner and stored in register C. Now the time interval of one minute is added to the previous minutes time. Then the minutes value is checked for being greater than or equal to sixty. If it isn't greater than or equal to sixty program execution jumps to STOR-TIME routine which converts time back into ASCII form to be printed as part of the output record. If the minutes value is greater than or equal to sixty it is subtracted by sixty to get the correct minutes value and one is added to the hours value.

Then the new hours value is checked for being thirteen which should be a one. If the hours value is thirteen,it is changed to one,which it should be,and program execution jumps to the STOR-TIME routine. If the hours value isn't thirteen,program execution jumps to the CHECK-12 routine. The new hours value is checked for being twelve and if it isn't program execution jumps to STOR-TIME routine. If the new hours value is twelve the AM or PM designation and possibly the day of the month will have to be changed.

If time designation was A it is changed to P to indicate that time is changing from the first twelve hours of the day to the last twelve hours of the day. Then program execution jumps to the STOR-TIME routine. If time designation was P program execution jumps to the PM routine which handles this condition. The P is changed to an A which indicates a change from the last twelve hours of the day to the first twelve hours of the next day. Now the day of the month must be incremented so the second digit of the day of the month is brought from memory in ASCII form and incremented one. If this results in an ASCII 40 this indicates that the first digit of the day of the month must be incremented one. So the ASCII value for the first digit of the day of the month is incremented one and the second digit of the day of the month is reset to an ASCII 30 to indicate a zero for the second digit of the day of the month. After being incremented the first digit of the day of the month is stored back into memory in ASCII form. Then the second digit of the day of the month is stored back into memory in ASCII form by the STOR-DATE routine. If the second digit of the day of the month wasn't an ASCII 40 then the program execution jumped to the STOR-DATE routine to store the ASCII form of the second digit of the day of the month back into memory space as part of the output record.

After getting the correct date put into the output record memory space the STOR-TIME routine is executed. This routine puts the current time just calculated back into the output record memory space. The PM or AM designation is put back into proper memory space. The first digit of minute time is converted back into ASCII form and stored in register D, and the second digit of minutes time is converted back into ASCII form and stored in register E. Then the minutes time in ASCII form in register pair DE is stored back into output record memory space. The same procedure is used with registers B and C to store hours value of time back into correct memory space. This completes the programming required to update the time and date.

Now the microprocessor is ready to wait until the interrupt occurs signaling the end of one minute delay. During this time an input to instruct the moisture content monitoring system to go into the WAIT mode can be entered. In the WAIT mode the system only does timekeeping to maintain the correct time and date. It is advantageous to put the system into the WAIT mode if the grain transfer equipment must be stopped such as for minor repair or to change the bin that grain is being removed from or being loaded into.

A WAIT-CHECK routine checks for wait input being entered. The microprocessor enters a subroutine to wait for an input. If there is no input sent from keyboard of DEC-Writer the microprocessor stays in the CIN subroutine until the interrupt occurs. If an input is entered from the keyboard of the DEC-WRITER it is brought into the microprocessor and then echoed back to the DEC-Writer to show it was received. The data entered is checked for being a W indicating system should go into WAIT mode. If data entered isn't a W, program execution jumps to RUN routine to check if it is a R indicating RUN mode is desired. If data entered was a W the direction indicator is pulled off the stack, the previous mode indicator is pulled off the stack, the W is pushed onto the stack as the mode indicator, and the direction indicator is pushed back onto the stack. Then a statement saying the system is in the WAIT mode is printed at the DEC-Writer. Program execution then returns to WAIT-CHECK routine to check for a new input changing the mode of the system or to await interrupt signal.

The RUN routine checks input for being a R which signals that the moisture content monitoring system should return to the RUN mode and start monitoring moisture content again. If the input isn't an R or a W then it is a incorrect input and program execution returns to WAIT-CHECK routine to await a correct input. If input is an R the Direction indicator is pulled off the stack, the previous mode indicator is pulled off the stack,

the new mode indicator, R, is pushed onto the stack, and the direction indicator is pushed back onto the stack. The mode indicator is put onto the stack so that after an interrupt occurs the system can decide to return to normal monitoring operation or only do timekeeping. A message stating system is in the RUN mode is then printed at the DEC-Writer.

Subroutines - The following discussion covers the software contained in memory space 0C00 through 0FFF. This is the subroutine software which is in Appendix B, directly following the main program software. Each subroutine has a NOP or two in the instruction column before it and these lines are used in the comment section to briefly describe what the subroutine does. This section will give a detailed discussion of the subroutines in the order in which they are stored in memory.

The CIN subroutine is used to input a character from the keyboard of the DEC-Writer into the CPU. This is done by checking the status word of the 8251 USART connected to Input Port 21. If the next to Least Significant line is low the microprocessor keeps checking the USART status until a high is received on this line. When a high on this line which is the RxRDY (Receiver Ready) line this indicates that the USART has a character to be input by the CPU. The character is brought into the accumulator by an input command for the data port of the USART which is Port 20. Now that the character has been put into the accumulator program execution is returned to the next address after the calling instruction in the main program.

Data is sent from the microprocessor to be printed at the DEC-Writer by the COUT subroutine. Data to be sent is stored in the accumulator when the COUT subroutine is called. In the COUT subroutine the data in the accumulator is saved on the stack. The status word is brought in by Input Port 21 and the Least Significant Bit which is the TxRDY (Transmitter Ready) status is checked until it goes high indicating that the transmitter data input register is empty and can be used to convert

parallel data into serial data and send the data character to the DEC-Writer. Then the data to be sent to the DEC-Writer is retrieved from the stack and sent to the USART by Output Port 20. The data is converted from parallel to serial and transmitted to the printer of the DEC-Writer and program execution is returned to the main program address immediately after the instruction which called the COUT subroutine.

The Bufout subroutine is used to print a block of data at the DEC-Writer. The main program provides register pair HL with the starting address of data to be printed and data in the following memory addresses is printed until a NOP (00) is found which signals the end of the data block to be transmitted. This subroutine is used to print the questions and statements contained in the moisture content monitoring system. Whenever the BUFOUT subroutine is entered whatever was in the accumulator is stored on the stack. The data stored in the memory address is loaded into the accumulator and printed by calling the COUT subroutine. Data continues to be printed by repeating this procedure until a NOP is found. Then what was originally in the accumulator and the flag status is restored from the stack and program execution is returned to the main program.

Data is brought in from the keyboard of the DEC-Writer and stored in ASCII form by the READ-DEC-ASCII subroutine. The starting address where data is to be stored is loaded into the HL register pair and data is stored in this address and the following address. Data is brought in from keyboard by CIN subroutine and echoed back to the printer to show the operator what he typed on the keyboard. Then the data is stored in ASCII form into the memory address in register pair HL and register pair HL is then incremented to next memory address. This procedure continues until a carriage return is received indicating end of data to be read and program execution returns to the main program.

The READ-DEC-ACT subroutine reads data from the keyboard and stores it in memory in actual form. This subroutine works in the same manner as the READ-DEC-ASCII subroutine except it converts data to actual form before storing it in memory. Conversion to actual form is done by substituting a zero in the first digit instead of a three and deleting the decimal point. This subroutine is only used to input numbers so all ASCII numbers have a three in the first digit. A 7 is represented as a hexadecimal 37 in ASCII and as a hexadecimal 07 in actual form.

Moisture content reading is read from the moisture meter and stored in memory in actual form by the READ-MC-ACT subroutine. The moisture content reading will be stored as a three byte value starting with the memory address loaded into register pair HL in the main program before calling this subroutine. The first digit of moisture content reading is read by Input Port 01 and stored directly into memory. The second and third digits are read by the Input Port 00 with the second digit in the four MSB (Most Significant Bit) and the first digit in the four LSB (Least Significant Bits). This value is converted into two bytes in actual form, stored into memory, and program execution then returns to main program.

The counter for readings interval between doing calculation for required moisture content of remaining grain is read from the keyboard by the READ-DEC-COUNT subroutine. This value can be either a one or two digit value. It is read from the keyboard by the CIN subroutine and printed at the printer on the DEC-Writer by the COUT subroutine. A carriage return signals the end of the input of data. The one or two digit value is converted into actual form and combined ,if two digits, and then stored into memory as a one byte value. If more than two digits are input the additional-digits are ignored and the microprocessor awaits a carriage return signaling end of input and after it is received program execution returns to main program.

The following subroutines are used with calculator card in doing cal-

culations required by the moisture content monitoring system. The RESET subroutine is used to reset calculator in preparation for making the next calculation. A RESET signal is sent by sending a low signal on the most significant line of output port 02. This line is held low for at least twenty five microseconds. Then a high signal is sent on this line to release the reset and program execution returns to main program.

Data and instructions are sent to the calculator by the LOAD-W-COUNT subroutine. Before calling this subroutine the main program initializes register pair BC to the first address where data to be sent is stored and loads register D with a counter value for number of data bytes to be sent to the calculator. The READY line is checked until it goes high indicating that calculator is ready to accept data. Data must be in calculator form so that it can be understood by the calculator. The data is then sent to the calculator through Output Port 03 along with a high on the STROBE-IN line which informs the calculator to read what is on the data lines. The STROBE-IN line is then cleared and the counter in register D is decremented. This procedure continues until the counter in register D is decremented to zero, then program execution returns to the main program.

Instructions telling the calculator that it will be read from and to have its data available to be read is done by the OUT subroutine. The OUT instruction is a two byte instruction and is sent to the calculator in the same way that the LOAD-W-COUNT subroutine sends data. The READY line is checked until it goes high, first part of OUT instruction is sent along with a high on STROBE-IN line to calculator, STROBE-IN line is cleared, READY line is checked until it goes high again, second part of OUT instruction is sent along with a high on STROBE-IN line to calculator, STROBE-IN line is cleared, and program execution is returned to main program. It doesn't matter what the second part of the OUT instruc-

tion is, so an hexadecimal 11 is used.

The READ-CAL subroutine reads the answer from the calculator and stores it in answer form in memory. When average moisture content value is calculated it is rounded to three significant digits by this subroutine. Before calling the subroutine the main program initializes register pair DE to the memory address where answer will be stored and a counter for number of bytes of answer to be read is loaded into register B. After entering the subroutine the counter in register B is saved in register C for making a check later in the subroutine. The DATA AVAILABLE* line is checked until it goes low indicating that calculator has data ready to be read. When the DATA AVAILABLE* line goes low,data is read from the calculator through Input Port 03, stored in memory in answer form, STROBE-OUT is set high, then cleared to prepare to read next byte of data, and register B counter is decremented. This procedure continues reading bytes of data until register B counter is zero. Then the original counter value is checked for being 05. If it isn't 05,then some calculation other than average moisture content was made so rounding isn't required and program execution is returned to main program. If 05 bytes were read this could be a calculation for average moisture content so it needs to be rounded.

To do the rounding an additional byte of data is read from the calculator. If this byte of data is less than five,no rounding is required so program execution jumps past rounding software and sends STROBE-OUT high signal to calculator, clears STROBE-OUT and returns program execution to main program. If data byte just read is greater than or equal to five, rounding is required. The third digit of average moisture content is recalled from memory, rounded up one, and stored back into memory. If the third digit is a decimal value program execution jumps to the CONTINUE 1 routine. If the third digit value was incremented from 09 to 0A then the third digit value is changed to 00 and stored back into cor-

rect memory space. Then the second digit value is incremented one and checked for being 0A. If it isn't 0A, then program execution jumps to CONTINUE 1 routine. If second digit value is 0A, it is changed to 00 and the first digit is incremented. If first digit isn't 0A, then program execution jumps to CONTINUE 1 routine. If first digit is 0A it is changed to 01 and the decimal point position indicator is decremented. For example a average moisture content value of 9.995 is read from calculator in answer form as 00 0B 09 09 09 05 and is rounded to a three digit answer value of 00 0A 01 00 00. The first byte is sign byte and the second byte is the decimal point position indicator. Now the CONTINUE 1 routine sends a STROBE-OUT high to calculator, clears it and program execution returns to main program.

All of the next eight subroutines except for the COPY subroutine are used to read data stored at one place in memory in one number form, convert data to another number form, and store converted data in another form at a new memory location. The COPY routine is for moving data from one block of memory space to another block of memory space without converting any of the data. The conversion subroutines are used to convert data that was input, read or calculated to a different number form to do comparison, new calculations, or to be printed at the DEC-Writer. Before the main program calls any of these subroutines it initializes register pair HL to the first address of the memory space where data is to be read, and register pair DE is initialized to the first address of the memory space where converted data will be stored. Also some of the subroutines use a counter for number of bytes to be converted. This counter is loaded into register B by the main program before the subroutine is called. Whenever the subroutine completes its task program execution is returned to the main program by the return instruction.

The COPY subroutine is used to copy the initial storage and calculation memory space from ROM (Read Only Memory) into RAM (Random Access Memory) where data can be written into this memory space. A data byte from the memory addressed by register pair HL is loaded into the accumulator and then stored into memory addressed by register pair DE. Both register pairs are incremented to the next memory address and the procedure continues until desired number of data bytes is transferred at which time program execution is returned to main program.

ASCII form numbers are converted into calculator form numbers by the ASCII-CAL subroutine. This subroutine is used to convert predicted number of readings which is input in ASCII form into calculator form for making calculations. A decimal 5 is a ASCII 35 and is a 05 in calculator form. Also the ASCII decimal point (2E) is converted into the calculator form decimal point (OA). The number of data bytes to be converted is controlled by the counter in Register B.

The ANS-CAL subroutine converts data from answer form as read from calculator into calculator form which can be used to make more calculations or be converted again into another form. The sign byte of answer form data is deleted since all values calculated are positive. A check is made to determine what the decimal point position indicator is and depending on the value of the decimal point position indicator a counter value is loaded into register C to show where to insert calculator form decimal point. After determining the value of the counter to be put into register C a data byte is converted from answer form into calculator form and registers B and C are decremented. This continues until register C is decrememted to zero. Then a calculator form decimal point is inserted into calculator form memory space and register B is decremented. Then data conversion continues until register B is zero indicating all data

conversion has been completed.

Actual form individual moisture content reading which was used in comparison tests is converted into ASCII form value to be stored in the output record by the ACT-ASCII subroutine. This subroutine doesn't use a counter for number of bytes to be converted. The individual moisture content reading has only three digits which are stored as three data bytes in actual form. The first two digits are converted from actual form into ASCII form. An ASCII decimal point is inserted into the ASCII memory space and the third digit is then converted from actual form into ASCII form. This produces a moisture content value composed of two digits, a decimal point, and a tenths digit.

The ACT-CAL subroutine converts actual form moisture content values into calculator form to be used to do more calculations. This subroutine operates exactly the same as the previous subroutine except the decimal point inserted is a calculator form decimal point.

Average moisture content values are converted from calculator form into ASCII form to be stored in the output record by the CAL-ASCII subroutine. This subroutine does use a counter to control the number of bytes to be converted. A check must be done on each byte to determine if it is the calculator form decimal point. If it is the calculator form decimal point, it is converted into the ASCII decimal point and stored in ASCII memory space. If it isn't the calculator form decimal point the number contained in the byte is converted from calculator form into ASCII form. This procedure continues until the counter in register B is decremented signaling all data bytes have been converted.

The CAL-ACT subroutine converts the calculator form of average moisture content into actual form so comparison tests can be ran. To do this the byte that contains the calculator form decimal point must be identified

and deleted. Other than the decimal point the calculator form and actual form are the same, so the remaining calculator form bytes are stored into actual form memory space.

Whenever an average moisture content value is calculated it must be converted from answer form as read from calculator into calculator form as two digits, a decimal, and a tenths digit. The AVG-ANS-CAL subroutine does this conversion. This requires determining if the decimal point position indicator byte contains an 0A or an 0B. If the decimal point position indicator is 0A this indicates that when the answer form is converted to calculator form the decimal point will be correctly placed between the second and third digit. The number bytes in answer form are the same as in calculator form. So whenever an 0A is in the decimal point indicator byte, two answer form bytes are stored as calculator form bytes, a calculator decimal point is inserted and the third answer form byte is stored as a calculator form byte. When the decimal point position indicator contains an 0B this indicates that there is only one digit before where the decimal should be placed. This occurrs when the average moisture content value that is calculated is less than 10.0 percent. In order to represent the correct amount of accuracy and to keep average moisture content value digits in correct position to make comparison tests there must be two digits, a decimal, and a tenths digit in the average moisture content value. So in this case a zero must be inserted as the first digit, the first digit from answer form is stored as the second digit in calculator form, and the second digit from answer form is rounded and stored as the third digit in calculator form. The rounding routine is similar to the one used to do the rounding in the READ-CAL subroutine.

The DECIMAL-HEX subroutine changes the one byte two digit decimal counter value for number of readings interval between calculation of requir-

ed moisture content for remaining grain into a one byte two digit hexadecimal value. The counter value must be changed from decimal to hexadecimal because this is the number system in which the microprocessor operates. Register pairs HL and DE are initialized by the main program to the memory addressess where the decimal value is stored and where the hexadecimal value will be stored. The subroutine loads the decimal value into the accumulator. Register C is loaded with the first digit of the decimal value as a counter for number of times to go through decrement cycle. The decrement cycle decrements the decimal value six times and register C is decremented. This procedure continues until the counter in register C is zero indicating the decimal value has been converted into a hexadecimal value. The hexadecimal value is then stored in the memory space addressed by register pair DE.

The INDIV-CHECK subroutine checks the individual moisture content reading limits that were input from the keyboard to determine if the high limit is greater than or equal to the low limit. If there was an error in the inputs from the keyboard this subroutine would help detect it in many cases. Register pair HL is loaded with the starting address where the high limit for individual moisture content reading is stored in actual form and register pair DE is loaded with the starting address where the low limit for individual moisture content reading is stored in actual form, and Register B is loaded with a counter for number of bytes to be compared. The high limit digit is loaded into register C and the low limit digit is loaded into the accumulator and the digits are compared. If low limit digit is less than high limit digit the low limit value is less than high limit value so program execution jumps to HI-AVG-READ in main program. If the low limit digit isn't less than or equal to high limit digit then the low limit value is greater than the high limit value so program execu-

tion jumps to READ-LIM-MESS to print a message stating this has occurred. The counter is decremented and checked for being zero. If the counter isn't zero the procedure is repeated until desired number of bytes have been compared or one of the previous cases occur.

A check similar to the INDIV-CHECK is performed by the AVG-CHECK to determine if high average moisture content limit is greater than or equal to low average moisture content limit. This subroutine operates the same except if inputs are correct program execution jumps to DIRECTION routine in main program and if inputs are incorrect program execution jumps to AVG-LIM-MESS to print message stating inputs are incorrect.

The DESIR-AVG-TEST subroutine checks to determine if the desired average moisture content value input at the keyboard is within the high and low limits for average moisture content. This subroutine operates by checking if high average moisture content limit is greater than or equal to desired average moisture content value and if low average moisture content limit is less than or equal to desired average moisture content value. The checks in this subroutine are done in the same manner as the checks in the previous subroutines.

When the individual moisture content reading is read from the moisture meter the HI-VALID-TEST and the LO-VALID-TEST routine check to make sure the reading is valid. The high valid moisture content reading is 50.0 percent and the low valid moisture content reading is 05.0 percent. These checks are performed the same way as the other checks. If a grain sample wasn't gotten into the test cell of the moisture meter a value in the high nineties will be given for corn or soybeans and a value between zero and one will be given for oats. These checks are used to detect if this occurs, if so the reading will be ignored.

The next four subroutines are HI-INDIV-TEST, LO-INDIV-TEST, HI-AVG-TEST, and LO-AVG-TEST. The first two check individual moisture content reading for being greater than or less than the limits input for individual moisture content reading. And the last two check average moisture content value for being greater than or less than the limits input for average moisture content. These subroutines work the same as the other comparison subroutines. Whenever a value is outside the limit being checked a message stating this is printed at the DEC-Writer.

A check is performed by the CHECK-READING subroutine when the system is in the load monitoring situation to determine if the predicted number of readings have been taken. Unlike the other comparisons which were done with data in actual form this subroutine works with data in calculator form. The number of readings remaining data is checked. The predicted number of readings have been taken if all of the data bytes except the decimal point byte contain zeros. When this occurs the direction of grain indicator on the stack is changed from N to Y to signal there will be no more calculations for required moisture content of remaining grain. A message stating that predicted number of readings have been made is printed and program execution jumps to the UNLOAD-GRAIN routine in main program.

The next nine subroutines are used to print messages from the comparison subroutines stating that a certain situation has occurred. These subroutines load the starting address of their particular message into register pair HL. Then the message is printed at the DEC-Writer using the BUFOUT subroutine. Finally the subroutine sends program execution to the correct address in the software.

At addresses 1580-1582 the instructions are stored which are sent to the calculator after it has been reset. Two NOPs should be sent after a reset and then the Master Clear (MCLR) instruction is sent to instruct cal-

culator to clear all registers, automatically set mantissa digit count to eight, and input and output values in floating point mode.

When an interrupt occurs at the end of the one minute time delay the microprocessor's program execution is sent to address 0038 by the RST 7 instruction. At this address the microprocessor is instructed to jump to address 1585 which is the Beginning of the INTR-RESET routine. This routine reloads counter zero of the 8253 Programmable Interval Timer to set up the one minute time delay again. Counter zero is reloaded in the same manner that was used to load it earlier in the main program software. The return address showing where program execution was when the interrupt occurred is pulled off of the stack. Then the return from subroutine address which is next on the stack is removed. These addresses are removed because they aren't going to be used and they must be removed after each interrupt to prevent them from building up on the stack and eventually moving down into RAM calculation space. Next, there is a interrupt enable command to allow interrupt to occur at end of one minute time delay. The system mode indicator is moved into the accumulator to check for WAIT or RUN mode. If the system is in the WAIT mode program execution is directed to the time-keeping routine to update the time. If the system is in the RUN mode program execution is directed to load a fresh sample and repeat monitoring operation.

Performance Tests

A test to simulate the use of the moisture monitoring system under actual conditions was performed to determine if the system would operate correctly. The system was operated with two different test procedures. One test procedure was simulating incoming grain being unloaded into a bin at a grain terminal. The other test procedure simulates grain being loaded onto some type of carrier to be transported to another grain terminal.

Approximately 25 kg of corn was obtained to prepare samples for testing the system. This corn was measured by the moisture meter to have an initial moisture content of 12.7 percent wet basis. Five groups of corn were prepared by adding varying amounts of water to each group. Three larger groups of 7.5 kg of corn were prepared with approximately one percent moisture content increments between each group. The other two groups of 1.25 kg of corn were prepared with one group being distinctly lower and the other group being distinctly higher in moisture content than the middle three groups. The approximate moisture content of the groups of corn was 13, 15, 16, 17, and 19 percent. All of the corn was prepared at the same time and a period of 24 hours was allowed for the moisture content in each group to approach an equilibrium value. During the waiting period the corn was manually stirred several times to keep damper corn mixed with dryer corn. This speeded the process of bringing moisture content towards an equilbrium value in each lot of corn. The moisture content for samples from each group will still vary some but this is appropriate since this will better simulate different moisture content of corn samples being pulled from corn arriving at a grain terminal.

The first test procedure was to simulate incoming corn at a grain terminal. An experimental design was setup to measure moisture content of samples drawn from the prepared groups in a random sequence. The test was to be two hours in length using one minute sampling intervals. So a random number table was used to generate 120 random numbers. The values were selected by randomly picking a value and taking it and the next 119 values which range from 0 to 9. When there is a 1, 2, or 3 a sample from the 15% moisture content corn is pulled and put in the moisture meter. When there is a 4, 5, or 6 a sample from the 16% moisture content corn is used and when there is a 7, 8, or 9 a sample from the

17% moisture content corn is used. The 13% or 19% moisture content corn is used whenever a zero appears. Whether the 13% or 19% moisture content corn is used depends on the list of random numbers preceding the zero. If a one is closer to the zero than a nine, then a sample from the 13% moisture content corn is used. If a nine is closer to the zero than a one, then a sample from the 19% moisture content corn is used. This is the random sequence used to load samples of varying moisture content into the moisture meter.

The inputs that the system asks for in preparation for the monitoring operation are then loaded by the operator. Upper and lower limits for an individual moisture content reading are 17.5 and 14.5. The three middle moisture content groups of corn are inside these limits and the high and low moisture content groups are outside these limits. Average moisture content for all groups of the corn should be approximately 16% so the upper and lower limits for average moisture content is set at 0.5% on either side of this value at 16.5% and 15.5%.

Then the monitoring process began with a corn sample being delivered to the moisture meter. The system loaded the sample into the moisture meter, a reading was taken, an output record was printed, and the sample was dumped. The operator then returned this sample to the corn group it was pulled from, delivered a fresh sample to the moisture meter, and the system repeated the monitoring procedure. Samples were loaded in the order that the corresponding random numbers were drawn. A printout of the output from the system during the test is shown in appendix D. The fourth reading is invalid because only a small fraction of the correct sample size was loaded and so an excessively low reading was obtained. Also, the fifth reading was invalid because no corn was loaded and an excessively high reading was obtained. This shows the system's capability to check for these type of invalid

readings and not use them in the calculation of average moisture content. Immediately after the 4:06 reading the system was put into the wait mode so that all it does is keep the correct time. This allows an interruption in the monitoring process for equipment repair or any other temporary delay. Then the system was put back into the run mode and the remainder of the test was completed.

During the test procedure for the unloading of incoming grain the moisture content monitoring system performed correctly. Timekeeping for the one minute sampling intervals and for the total test period was correct. Calculation of average moisture content with appropriate rounding was correctly done. All of the checks on limits and the appropriate output messages operated as required. So the system worked as expected and performed all requirements during this phase of testing.

The other test was run to determine how well the moisture content monitoring system performed under conditions where grain was being loaded onto a carrier to be transported to another grain terminal. In this test only two of the groups of corn were used. These groups contained corn of 15% and 16% moisture content. This test used these groups of corn in an attempt to obtain a shipment of corn with a moisture content of 15.4 percent. Moisture content of corn must be between 14.0 and 15.5 percent to be #2 grade corn so if 15.4 percent is obtained this corn shipment would qualify for #2 grade.

Inputs required by the system are set by the operator. Upper and lower limits for individual moisture content readings are set as 16.2 and 14.8 percent so almost all of the corn from the two moisture content groups used should be within the limits. Upper and lower limits for average moisture content are set at 15.8 and 15.0 percent to be 0.4 percent on either side of the desired final average moisture content.

The desired final average moisture content is 15.4 percent. There should be 120 readings with a 10 reading interval between each calculation of what the remaining moisture content should be to obtain the desired final average moisture content.

The monitoring procedure is started using samples pulled from the corn group of approximately 15 percent moisture content. After each ten readings output is printed showing what the average moisture content of the remaining grain should be to obtain the desired final average moisture content. Samples from the 15 percent moisture content group are used until this output states that the remainder of the grain should have an average moisture content of approximately 16 percent. After seventy readings have been taken, the output states that the remainder of the grain should have an average moisture content of 16.1 percent. So the system is put into the wait mode which simulates the actual waiting at a grain terminal to switch to pulling grain from a different bin. Then the system is put into the run mode and samples pulled from the 16 percent moisture content group are used. This continues until all 120 readings have been taken and a message is printed stating the predicted number of readings have been taken. Several more readings are taken to show that the system stops calculating and outputting information concerning the average moisture content of remaining grain. This completed the test procedure and the moisture content monitoring system was then stopped.

During this test procedure the system performed well with all outputs and calculations being correct. The second reading after the wait mode is lower than what it should be. When the solenoid opened the trap door at the top of the test cell to load the previous sample the spring on the trap door didn't pull the trap door completely closed.

So on the next sample, part of sample escaped into the test cell too early and when the remainder of the sample was loaded, an erroneous reading was obtained. This was the only time during both test procedures that this situation occurred. If this became a problem a slightly stronger spring could be installed on the trap door to prevent this situation from occurring. All other parts of the system functioned correctly during this test.

In summarizing the results of the test the moisture content monitoring system worked well throughout both tests. All calculations and rounding of values were done correctly. All of the checks worked and output information was printed at the appropriate time. These tests show that the system will perform the tasks for which it was designed.

APPENDIX A

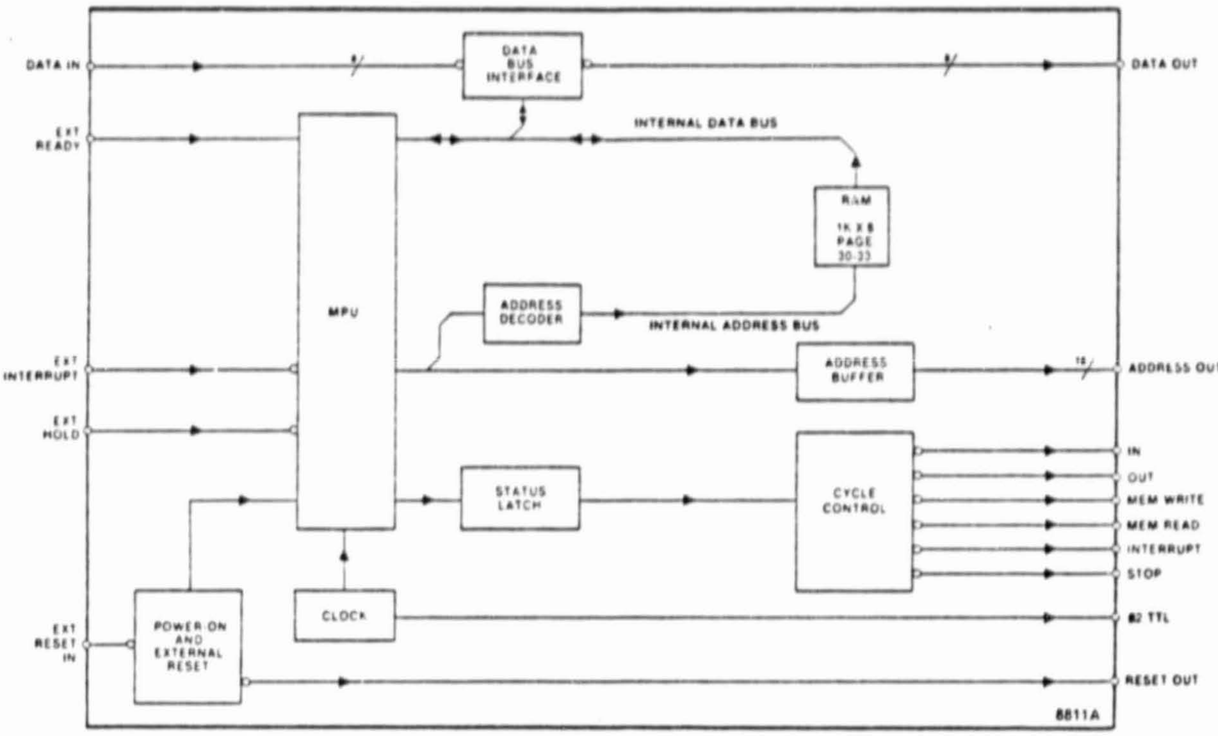
WIRING SCHEMATICS FOR ALL CARDS USED AND BACKPLANE WIRING FOR ALL INTERFACE CONNECTIONS

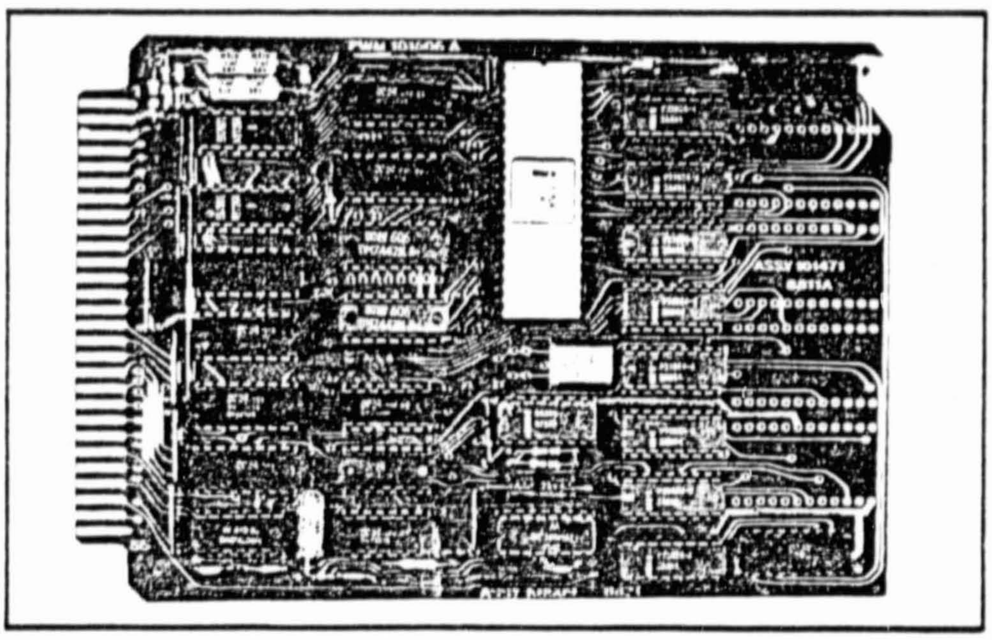
This appendix contains the wiring schematics for all of the cards used in the system, both those that were bought and the one that was wirewrapped within the department. Also, contained are the backplane wiring list for the USART and timer card, calculator card, triac card, and the moisture meter printer port.

The 8811A is a printed circuit card which implements the 8 bit 8080A Microprocessor as a fully TTL buffered microprocessor with clock, reset and 3-state data and address busses. The card also includes 1K bytes of RAM external memory control and I/O control.

FEATURES

- 8080A Processor
- Separate 8 bit data bus in and out
- 16 address lines (65K bytes of memory address)
- 3-state data, address and memory control for DMA
- Crystal clock
- Power-on and external reset
- 1024 bytes of 8 bit RAM
- 1 u sec time state





8811A PROCESSOR CARD

CARD DIMENSIONS

- 4.50 in. (11.43 cm) high by 6.50 in. (16.51 cm) long
- 0.48 in. (1.22 cm) maximum profile thickness
- 0.062 in. (0.16 cm) printed circuit board thickness

CARD INCLUDES

- Card ejector
- One 8080A Processor
- 1K 8 bit bytes of 2102 RAM
- Crystal clock circuit
- Power-on and external reset

INSTRUCTION EXECUTION CAPABILITY

- Executes all of the 8080A Processor instructions
- 1.00 microseconds time state cycle ± 0.005%
- Instructions require from 4 to 18 time states

MEMORY

- Maximum access time: 1.00 microseconds
- PROM memory: 1702A or equivalent
- RAM memory: 2102-4 or equivalent

INPUTS (Active low logic, loading 1 TTL load, except where noted)

- 8 Data lines
- 1 Memory ready control, active high
- 1 Interrupt request
- 1 Reset control
- 1 Hold control

8811A EDGE CONNECTOR PIN LIST

SIGNAL	SIGNAL FLOW	PIN NUMBER	PIN NUMBER	SIGNAL FLOW	SIGNAL
+5 VOLTS	IN	2	1	IN	+5 VOLTS
GROUND	IN	4	3	IN	GROUND
-5 VOLTS	IN	6	5	IN	-5 VOLTS
DIN 8*	IN	8	7	OUT	DOUT 8*
DIN 7*	IN	10	9	OUT	DOUT 7*
DIN 6*	IN	12	11	OUT	DOUT 6*
DIN 5*	IN	14	13	OUT	DOUT 5*
DIN 4*	IN	16	15	OUT	DOUT 4*
DIN 3*	IN	18	17	OUT	DOUT 3*
DIN 2*	IN	20	19	OUT	DOUT 2*
DIN 1*	IN	22	21	OUT	DOUT 1*
A16	OUT	24	23	OUT	A8
A15	OUT	26	25	OUT	A7
A14	OUT	28	27	OUT	A6
A13	OUT	30	29	OUT	A5
A12	OUT	32	31	OUT	A4
A11	OUT	34	33	OUT	A3
A10	OUT	36	35	OUT	A2
A9	OUT	38	37	OUT	A1
STOP*	OUT	40	39	OUT	WAIT*
IN*	OUT	42	41	OUT	OUT*
SPARE	OUT	44	43	OUT	WRM*
RDM*	OUT	46	45	OUT	INTR*
HLTA*	OUT	48	47	IN	IREQ*
HLD*	IN	50	49	IN	RDY
RESET*	IN	52	51	OUT	TTL Ø2
SYNC*	OUT	54	53	OUT	RST*
+12 VOLTS	IN	56	55	IN	+12 VOLTS

*Designates active low level logic

OUTPUTS (Active low logic, drive capability 10 TTL loads except where noted)

The designated output lines go to the 3-state condition in response to the HOLD input.

- 8 Data lines, 8 TTL loads, 3-state
- 16 Address lines, 20 TTL loads active high, 3-state
- 1 Memory Write Clock, 3-state with pull-up
- 1 Memory Read, 3-state
- 1 I/O Ouput Clock, 3-state with pull-up
- 1 I/O Input, 3-state
- 1 Interrupt Response, 3-state
- 1 Wait Control, 3-state
- 1 Stop Control
- 1 Reset Control
- 1 Sync Pulse Line
- 1 Clock Pulse Line, 8 TTL loads
- 1 SPARE: Write Out (WO), Oscillator (OSC), Memory Cycle 1 (M1), STACK or Interrupt Enable (INTE) are jumper selectable on Printed Circuit Board

POWER REQUIREMENTS

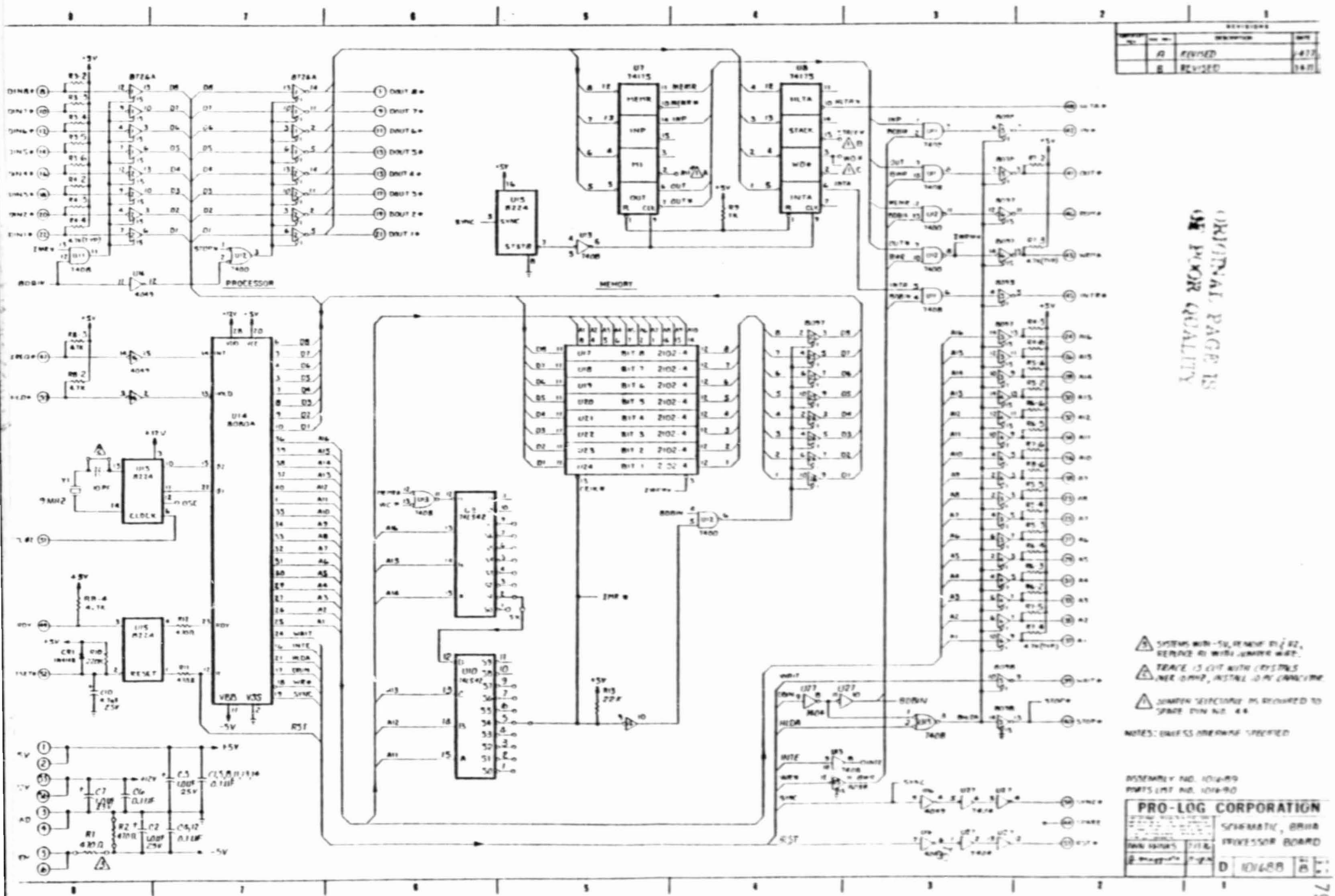
+VDD	=	+12 volts ±5% at 80 mA maximum
+VCC	=	+5 volts ±5% at 1.7 A maximum, fully loaded (50 mA per RAM)
GND	=	0 volts
-VBB	=	-5 volts ±5% at 7 mA maximum

OPERATING TEMPERATURE RANGE: 0-55°C

CONNECTOR REQUIREMENTS: 56 pin, 28 position dual-readout on 0.125 in. (0.318 cm) centers

PRO-LOG CORPORATION 2411 Garden Road Monterey, California 93940 Telephone (408) 372-4593 TWX:910-360-7082

Wiring schematic for 8811A processor card



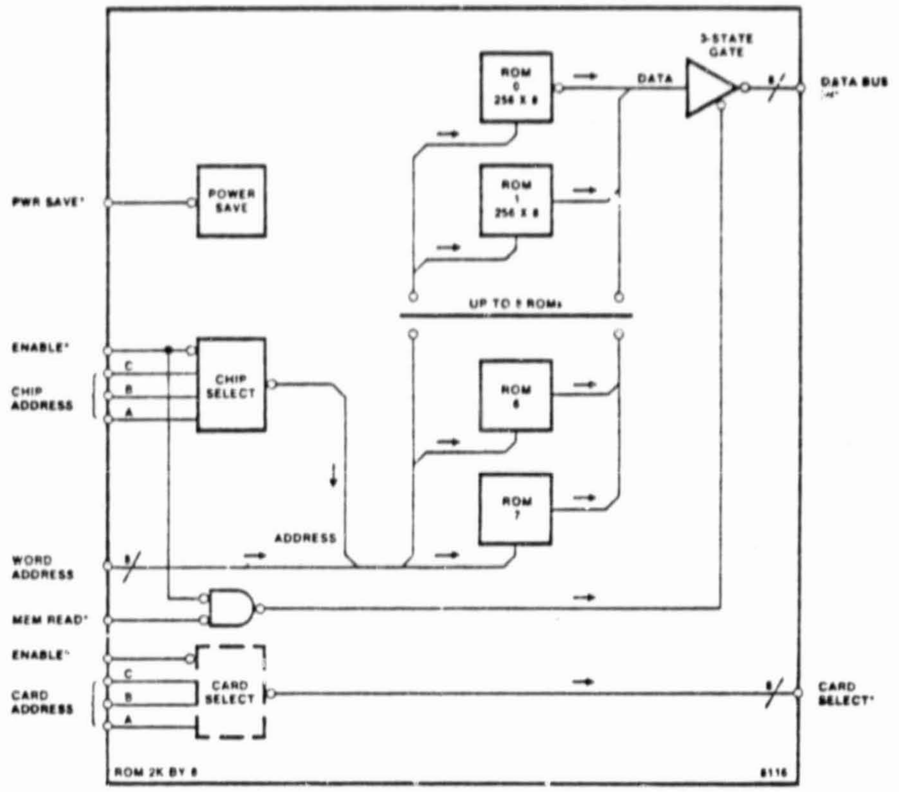
ORIGINAL PAGE IS OF POOR QUALITY

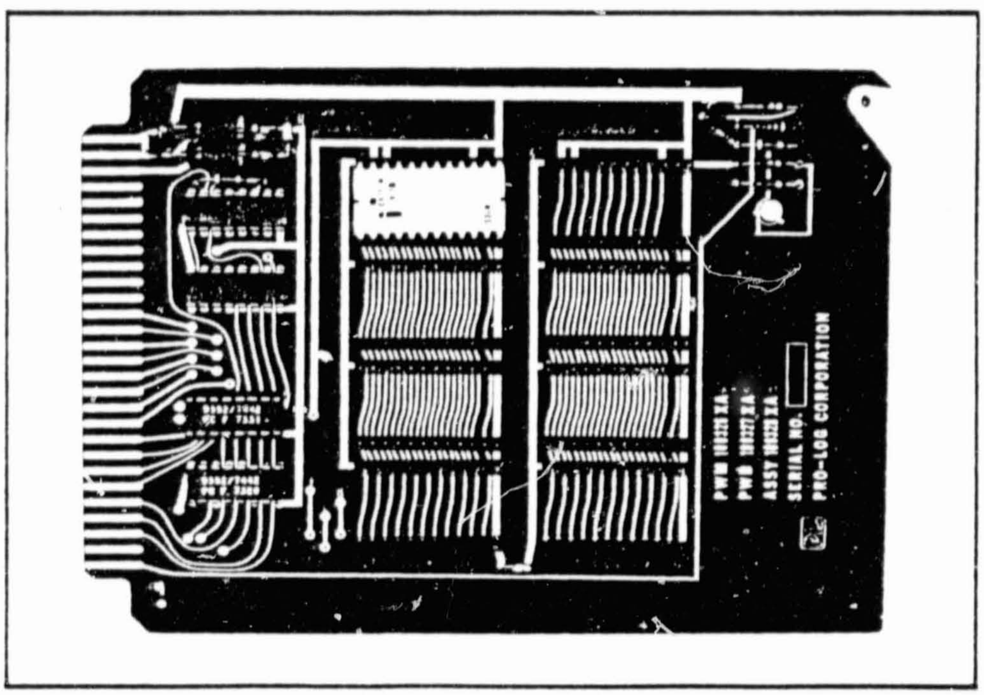
MPS SYSTEM COMPONENTS
8116 ROM CARD
2K BYTES BY 8-BITS

A printed circuit card which implements the 1302, 1602, or 1702 ROMs as read only memory for the MPS series of 8-bit microprocessors. The 8116 is organized as 2048 bytes of 8-bits.

FEATURES

- 2048 bytes of ROM memory capacity
- Card address allows sytem expansion to 16 K bytes
- Switches power for power-save on ROMs
- Sockets accept masked (1302) or programmable (1702) ROMs





8116 ROM

CARD DIMENSIONS

- 4.50 in. (11.43 cm) high by 6.50 in. (16.51 cm) long
- 0.48 in. (1.22 cm) maximum profile thickness
- 0.062 in. (0.16 cm) printed circuit board thickness

CARD INCLUDES

- Card ejector
- 8 ROM sockets
- Chip select circuits
- Card select option
- Power-save circuit

MAXIMUM CARD CAPABILITIES

- Eight 1302, 1602 or 1702 ROMs (2048 bytes of Read Only Memory)

INTERFACE

Inputs (Active low logic, loading 1 TTL load, except where noted)

- 8 word address lines, 8 MOS loads each, active high
- 3 chip address lines, active high
- 1 chip address enable, 2 TTL loads
- 3 card address, active high
- 1 card address enable
- 1 memory read control
- 1 power-save control, active high

Outputs (Active low logic, drive capability 10 TTL loads)

- 8 data lines, 3-state
- 8 card select lines

8116 EDGE CONNECTOR PIN LIST					
		PIN NUMBER	PIN NUMBER		
	SIGNAL FLOW			SIGNAL FLOW	
SIGNAL					SIGNAL
+5 VOLTS	IN	2	1	IN	+5 VOLTS
GROUND	IN	4	3	IN	GROUND
-10 VOLTS	IN	6	5	IN	-10 VOLTS
DIN8*	OUT	8	7		
DIN7*	OUT	10	9		
DIN6	OUT	12	11		
DIN5*	OUT	14	13		
DIN4*	OUT	16	15		
DIN3*	OUT	18	17		
DIN2*	OUT	20	19		
DIN1*	OUT	22	21		
		24	23	IN	WORD ADR (A8)
		26	25	IN	WORD ADR (A7)
CARD ADR-4 (A14)	IN	28	27	IN	WORD ADR (A6)
CARD ADR-2 (A13)	IN	30	29	IN	WORD ADR (A5)
CARD ADR-1 (A12)	IN	32	31	IN	WORD ADR (A4)
ROM ADR-4 (A11)	IN	34	33	IN	WORD ADR (A3)
ROM ADR-2 (A10)	IN	36	35	IN	WORD ADR (A2)
ROM ADR-1 (A9)	IN	38	37	IN	WORD ADR (A1)
CARD SEL-3*	OUT	40	39	OUT	CARD SEL 0*
CARD SEL-4*	OUT	42	41	IN	CARD ENABLE*
CARD SEL-5*	OUT	44	43	OUT	CARD SEL-1*
CARD SEL-6*	OUT	46	45	OUT	CARD SEL-2*
		48	47	IN	CHIP ENABLE*
		50	49	OUT	CARD SEL-7*
		52	51	OUT	PULL-UP
RDM*	IN	54	53		
		56	55	IN	POWER SAVE*

*Designates Active Low Level Logic

POWER REQUIREMENTS

+VCC = +5 volts ± @ 300 mA maximum fully loaded (35 mA per ROM)
GND = 0 volts
-VDD = −10 volts + @ 300 mA maximum fully loaded (35 mA per ROM)

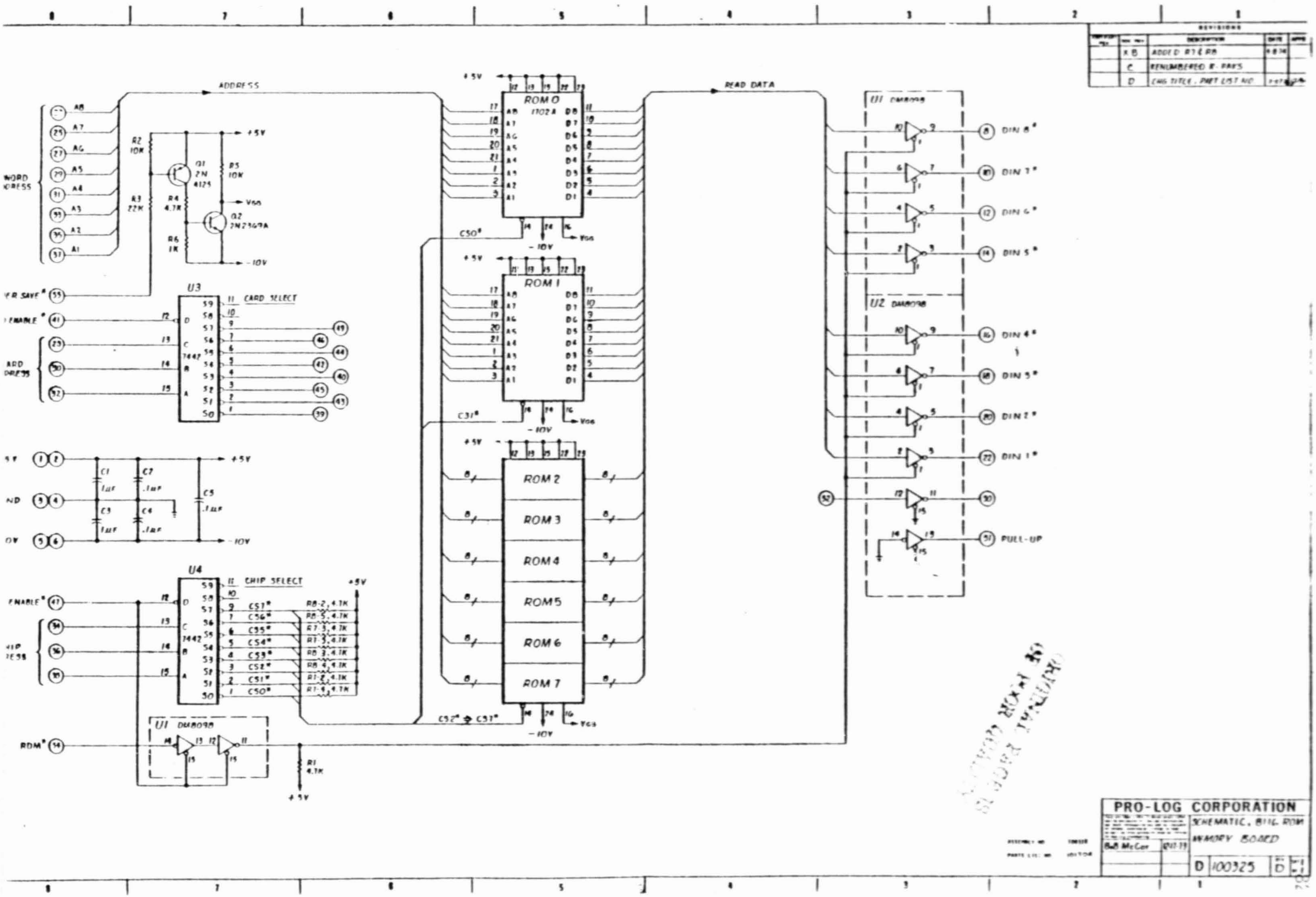
OPERATING TEMPERATURE RANGE: 0-55°C

CONNECTOR REQUIREMENTS: 56 pin, 28 position, dual-readout on 0.125 centers

APPLICATION NOTES

If only one memory card is used in the total system tie the Chip Enable input to ground. If more than one memory card is used in the system, at least one Card Select circuit must be used and the Chip Enable lines individually wired to the Card Select line for the desired card address.

PRO-LOG CORPORATION 2411 Garden Road Monterey, California 93940 Telephone (408) 372-4593 TWX: 910-360-7082

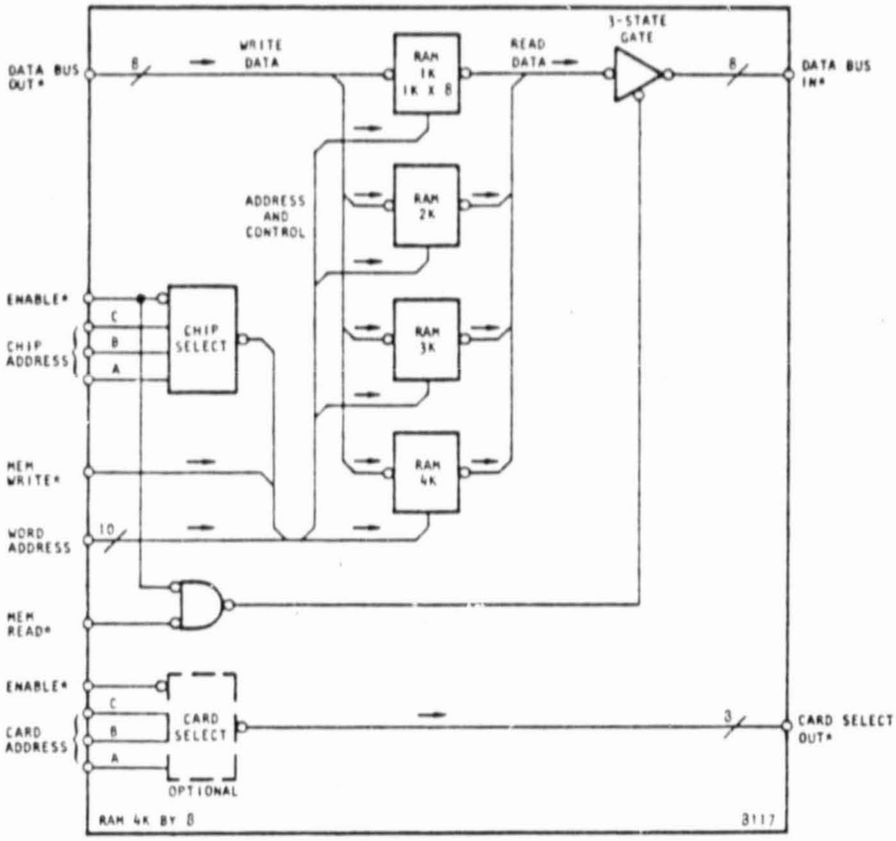

REVISIONS
ADDED R7 & R8
RENUMBERED R-PAKS
CHG TITLE, PART LIST NO.
ADDRESS
READ DATA
WORD ADDRESS
ROM 0
1702A
ROM 1
ROM 2
ROM 3
ROM 4
ROM5
ROM 6
ROM 7
U1 DM8098
U2 DM8098
DIN 8*
DIN 7*
DIN 6*
DIN 5*
DIN 4*
DIN 3*
DIN 2*
DIN 1*
PULL-UP
U3
CARD SELECT
U4
CHIP SELECT
Q1 2N 4125
Q2 2N2369A
CS0*
CS1*
CS2* ÷ CS7*
ROM*
ORIGINAL PAGE IS OF POOR QUALITY
PRO-LOG CORPORATION
SCHEMATIC, 8116 ROM MEMORY BOARD
D 100325

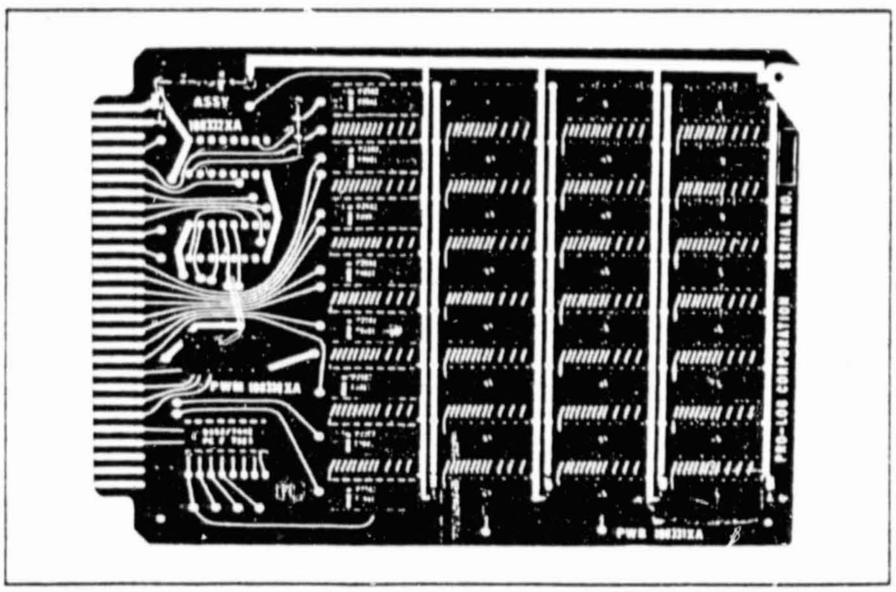
MPS SYSTEM COMPONENTS
8117 RAM
4K BYTES BY 8 BITS

A printed circuit card wh‹ch implements 2102-4 static MOS RAM as read-write-memory for the MPS series of 8-bit microprocessors. ‹he 8117 is organized as four 1024 byte increments and can be used as 1K, 2K, 3K, or 4K bytes of 8-bit memory.

FEATURES

- **4096 bytes of 8-bit RAM capacity**
- **Use as 1K, 2K, 3K, or 4K bytes of RAM**
- **Static MOS RAM**
- **Card addressing allows direct expansion to 64 K bytes**
- **3-State data bus output**





8117 RAM

CARD DIMENSIONS

- 4.50 in. (11.43 cm) high by 6.50 in. (16.51 cm) long
- 0.48 in. (1.22 cm) maximum profile thickness
- 0.062 in. (0.16 cm) printed circuit board thickness

CARD INCLUDES

- Card ejector
- 32 sockets
- Eight 2102-4 static RAM devices, 1024 bytes
- Chip select circuit.
- Socket for card select option

INTERFACE

Inputs
(Active low logic, loading 1 TTL load, except where noted)

- 8 Data lines
- 10 Word Address, Active high, 32 MOS loads
- 3 Chip Address, Active high
- 1 Chip Enable, 2 TTL loads
- 3 Card Address, Active high
- 1 Card Enable
- 1 Memory Read
- 1 Memory Write, 32 MOS loads

Outputs (Active low, drive capability 10 TTL loads)

- 8 Data lines, 3-state
- 8 Card Select lines

EDGE CONNECTOR PIN LIST

SIGNAL	SIGNAL FLOW	PIN NUMBER	PIN NUMBER	SIGNAL FLOW	SIGNAL
+5 VOLTS	IN	2	1	IN	+5 VOLTS
GROUND	IN	4	3	IN	GROUND
		6	5		
DIN8*	OUT	8	7	IN	DOUT-8*
DIN7*	OUT	10	9	IN	DOUT-7*
DIN6*	OUT	12	11	IN	DOUT-6*
DIN5*	OUT	14	13	IN	DOUT-5*
DIN4*	OUT	16	15	IN	DOUT-4*
DIN3*	OUT	18	17	IN	DOUT-3*
DIN2*	OUT	20	19	IN	DOUT-2*
DIN1*	OUT	22	21	IN	DOUT-1*
CARD ADR-4	IN	24	23	IN	WORD ADR (A8)
CARD ADR-2 (A14)	IN	26	25	IN	WORD ADR (A7)
CARD ADR-1 (A13)	IN	28	27	IN	WORD ADR (A6)
RAM ADR-4	IN	30	29	IN	WORD ADR (A5)
RAM ADR-2 (A12)	IN	32	31	IN	WORD ADR (A4)
RAM ADR-1 (A11)	IN	34	33	IN	WORD ADR (A3)
WORD ADR (A10)	IN	36	35	IN	WORD ADR (A2)
WORD ADR (A9)	IN	38	37	IN	WORD ADR (A1)
CARD SEL-3*	OUT	40	39	OUT	CARD SEL-0*
CARD SEL-4*	OUT	42	41	IN	CARD ENABLE*
CARD SEL-5*	OUT	44	43	OUT	CARD SEL-1*
CARD SEL-6*	OUT	46	45	OUT	CARD SEL-2*
		48	47	IN	CHIP ENABLE*
		50	49	OUT	CARD SEL-7*
		52	51		
RDM*	IN	54	53		
WRM*	IN	56	55		

8117

POWER REQUIREMENTS

+VCC = +5 volts ±5% @ 1.6 Amps maximum (50 mA per RAM)
GND = 0 volts

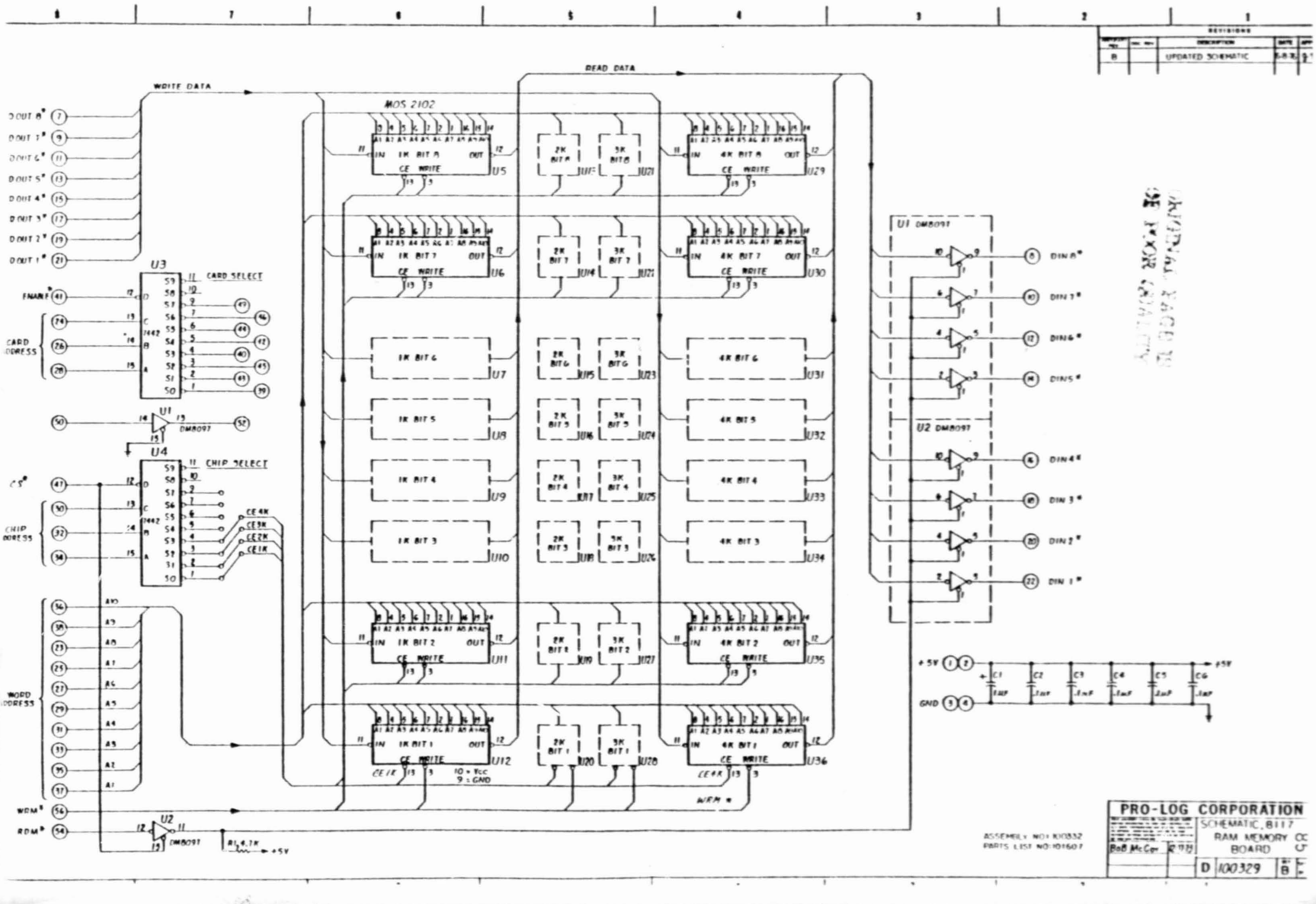
OPERATING TEMPERATURE RANGE: 0-55°C

CONNECTOR REQUIREMENTS: 56 pin, 28 position dual-readout on 0.125 in. (0.318 cm) centers

APPLICATION NOTES

If only one memory card is used in the total system, tie the Chip Enable input to ground. If more than one memory card is used in the system, at least one Card Select circuit must be used and the Chip Enable lines individually wired to the Card Select line for the desired card address.

PRO-LOG
CORPORATION 2411 Garden Road Monterey, California 93940 Telephone (408) 372-4593
TWX: 910-360-7082

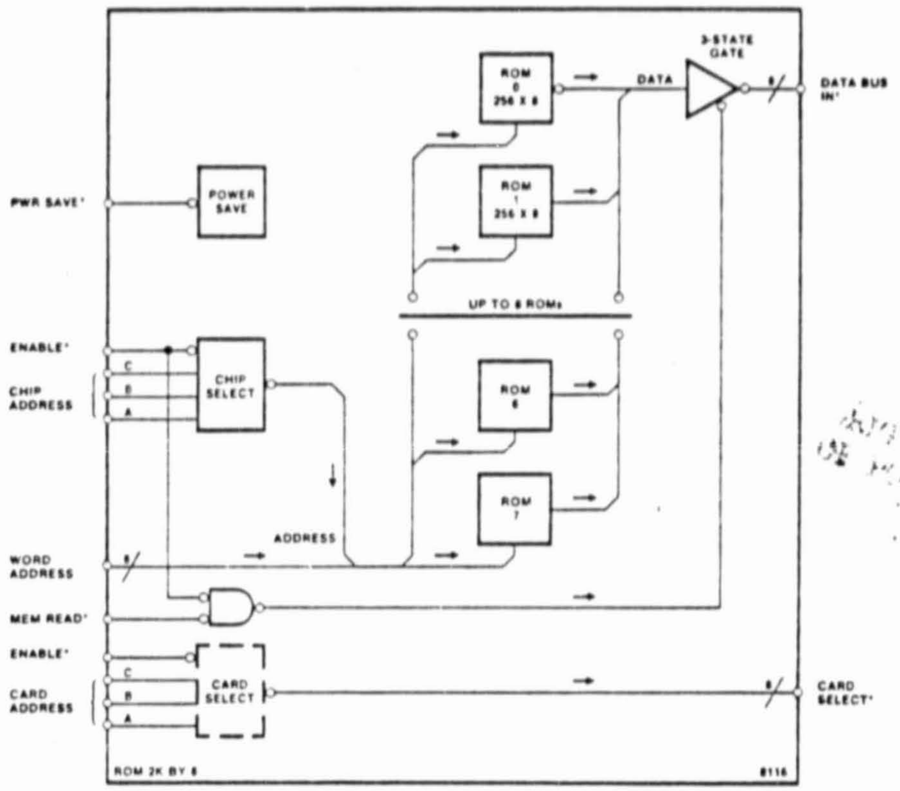

REVISIONS
B
UPDATED SCHEMATIC
WRITE DATA
READ DATA
MOS 2102
DOUT 8*
DOUT 7*
DOUT 6*
DOUT 5*
DOUT 4*
DOUT 3*
DOUT 2*
DOUT 1*
U3
CARD SELECT
CARD ADDRESS
U4
CHIP SELECT
CS*
CHIP ADDRESS
CE4K
CE3K
CE2K
CE1K
WORD ADDRESS
WRM*
RDM*
U1
U2
DM8097
R1 4.7K
+5V
1K BIT 8
1K BIT 7
1K BIT 6
1K BIT 5
1K BIT 4
1K BIT 3
1K BIT 2
1K BIT 1
2K BIT 8
3K BIT 8
4K BIT 8
4K BIT 7
4K BIT 6
4K BIT 5
4K BIT 4
4K BIT 3
4K BIT 2
4K BIT 1
CE WRITE
10 = Vcc
9 = GND
CE1K
CE4K
U1 DM8097
U2 DM8097
DIN 8*
DIN 7*
DIN 6*
DIN 5*
DIN 4*
DIN 3*
DIN 2*
DIN 1*
+5V
GND
ASSEMBLY NO: 100332
PARTS LIST NO: 101607
PRO-LOG CORPORATION
SCHEMATIC, 8117
RAM MEMORY CARD
BOARD
Bob McCor
D 100329 B

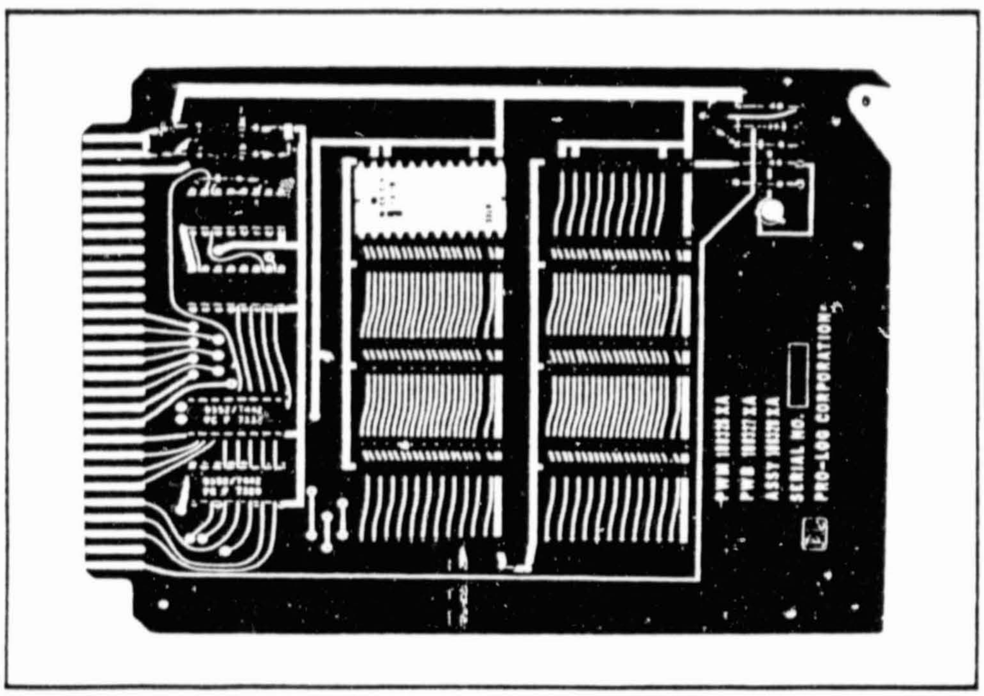
MPS SYSTEM COMPONENTS
8116 ROM CARD
2K BYTES BY 8-BITS

A printed circuit card which implements the 1302, 1602, or 1702 ROMs as read only memory for the MPS series of 8-bit microprocessors. The 8116 is organized as 2048 bytes of 8-bits.

FEATURES

- 2048 bytes of ROM memory capacity
- Card address allows sytem expansion to 16 K bytes
- Switches power for power-save on ROMs
- Sockets accept masked (1302) or programmable (1702) ROMs





8116 ROM

CARD DIMENSIONS

- 4.50 in. (11.43 cm) high by 6.50 in. (16.51 cm) long
- 0.48 in. (1.22 cm) maximum profile thickness
- 0.062 in. (0.16 cm) printed circuit board thickness

CARD INCLUDES

- Card ejector
- 8 ROM sockets
- Chip select circuits
- Card select option
- Power-save circuit

MAXIMUM CARD CAPABILITIES

- Eight 1302, 1602 or 1702 ROMs (2048 bytes of Read Only Memory)

INTERFACE

Inputs (Active low logic, loading 1 TTL load, except where noted)

- 8 word address lines, 8 MOS loads each, active high
- 3 chip address lines, active high
- 1 chip address enable, 2 TTL loads
- 3 card address, active high
- 1 card address enable
- 1 memory read control
- 1 power-save control, active high

Outputs (Active low logic, drive capability 10 TTL loads)

- 8 data lines, 3-state
- 8 card select lines

8116 EDGE CONNECTOR PIN LIST					
SIGNAL	SIGNAL FLOW	PIN NUMBER	PIN NUMBER	SIGNAL FLOW	SIGNAL
+5 VOLTS	IN	2	1	IN	+5 VOLTS
GROUND	IN	4	3	IN	GROUND
-10 VOLTS	IN	6	5	IN	-10 VOLTS
DIN8*	OUT	8	7		
DIN7*	OUT	10	9		
DIN6*	OUT	12	11		
DIN5*	OUT	14	13		
DIN4*	OUT	16	15		
DIN3*	OUT	18	17		
DIN2*	OUT	20	19		
DIN1*	OUT	22	21		
		24	23	IN	WORD ADR (A8)
		26	25	IN	WORD ADR (A7)
CARD ADR-4 (A14)	IN	28	27	IN	WORD ADR (A6)
CARD ADR-2 (A13)	IN	30	29	IN	WORD ADR (A5)
CARD ADR-1 (A12)	IN	32	31	IN	WORD ADR (A4)
ROM ADR-4 (A11)	IN	34	33	IN	WORD ADR (A3)
ROM ADR-2 (A10)	IN	36	35	IN	WORD ADR (A2)
ROM ADR-1 (A9)	IN	38	37	IN	WORD ADR (A1)
CARD SEL-3*	OUT	40	39	OUT	CARD SEL 0*
CARD SEL-4*	OUT	42	41	IN	CARD ENABLE*
CARD SEL-5*	OUT	44	43	OUT	CARD SEL-1*
CARD SEL-6*	OUT	46	45	OUT	CARD SEL-2*
		48	47	IN	CHIP ENABLE*
		50	49	OUT	CARD SEL-7*
		52	51	OUT	PULL-UP
RDM*	IN	54	53		
		56	55	IN	POWER SAVE*

*Designates Active Low Level Logic

POWER REQUIREMENTS

+VCC = +5 volts ± @ 300 mA maximum fully loaded (35 mA per ROM)
GND = 0 volts
-VDD = -10 volts + @ 300 mA maximu[illegible]ully loaded (35 mA per ROM)

OPERATING TEMPERATURE RANGE: [illegible]°C

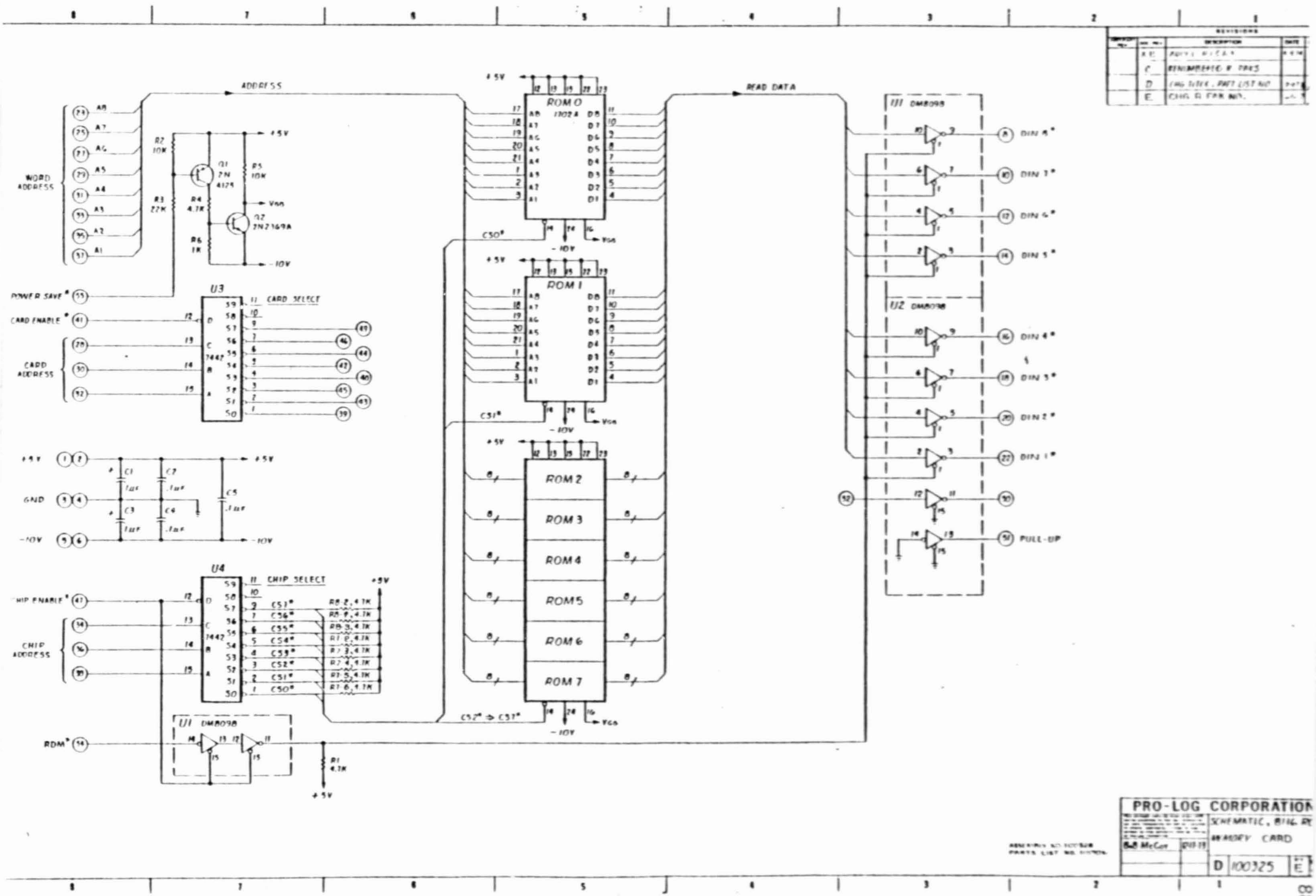
CONNECTOR REQUIREMENTS: [illegible]8 position, dual-readout on 0.125 centers

APPLICATION NOTES

If only one memory card is use[illegible]n the total system tie the Chip Enable input to ground. If more than one memory card is used in the system, at least one Card Select circuit must be used and the Chip Enable lines individually wired to the Card Select line for the desired card address.

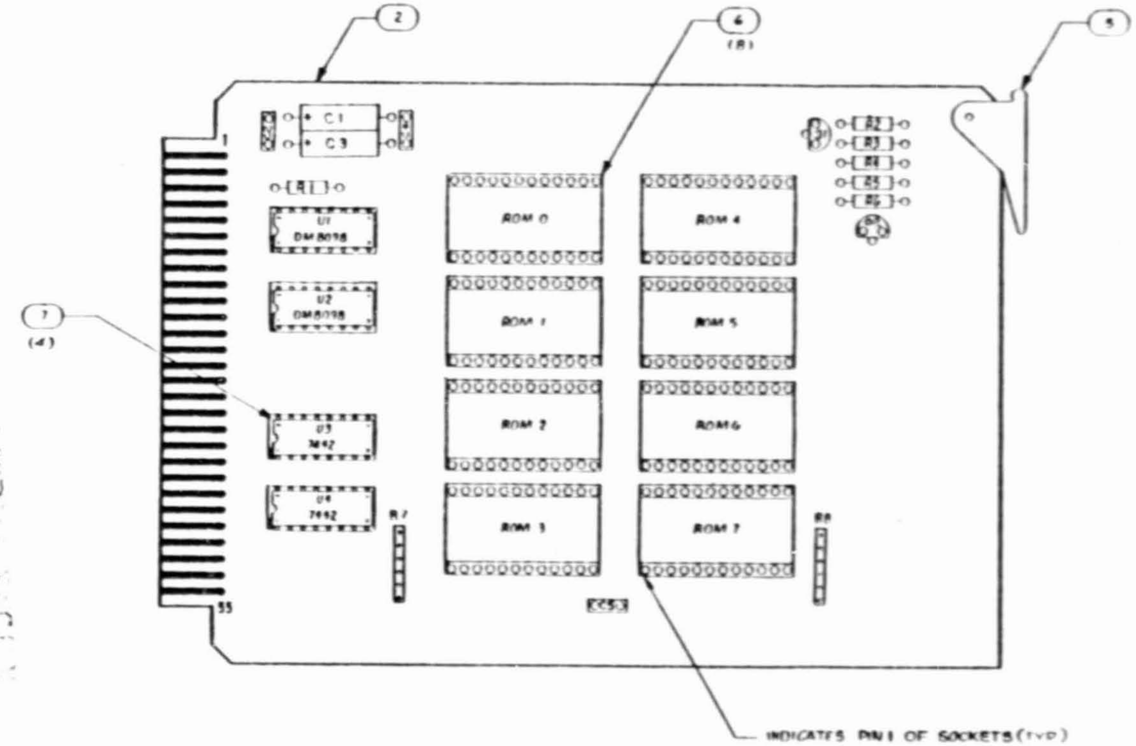
PRO-LOG CORPORATION 2411 Garden Road Monterey, California 93940 Telephone (408) 372-4593
TWX: 910-360-7082

Wiring schematic for 8116 ROM card



REVISIONS

	REV	DESCRIPTION	DATE
	XB	ADDED R7, R8	[illegible]
	C	RENUMBERED R PAKS	
	D	ADDED SOCKETS	[illegible]
	E	MOVED R PAKS, AND Q2	



ITEM	DESCRIPTION	REF DESIGNATION
21	NETWORK, 4.7K	R7, R8
20	4.7K CC ¼W	R1, R4
19	22K	R3
18	1K	R6
17	10K CC ¼W	R2, R5
15	2N4125 TO-92	Q1
14	2N2369A TO-18	Q2
9	.1µF 50V	C2, C4, C5
8	1µF 25V	C1, C3

PRO-LOG CORPORATION

ASSY, 8116 ROM MEMORY CARD

ROB McCOY 12-20-75

D 100328 E

SCHEMATIC NO. 100325
PARTS LIST NO. 101704

1. FOR ASSEMBLY PROCEDURES SEE AS1004.

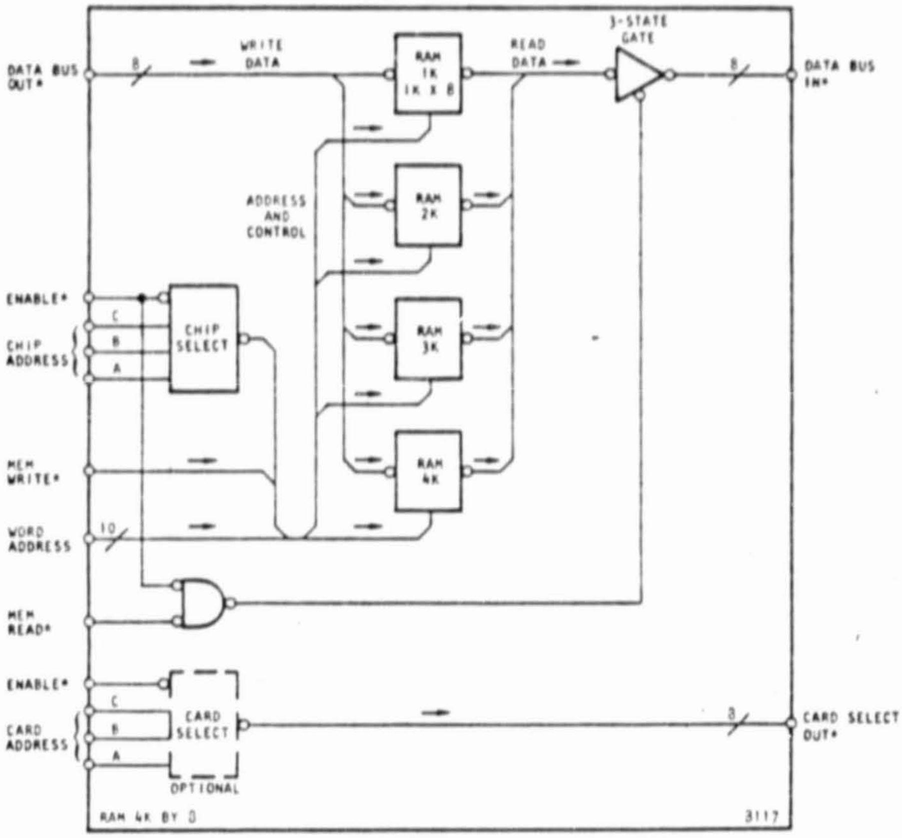
NOTES:

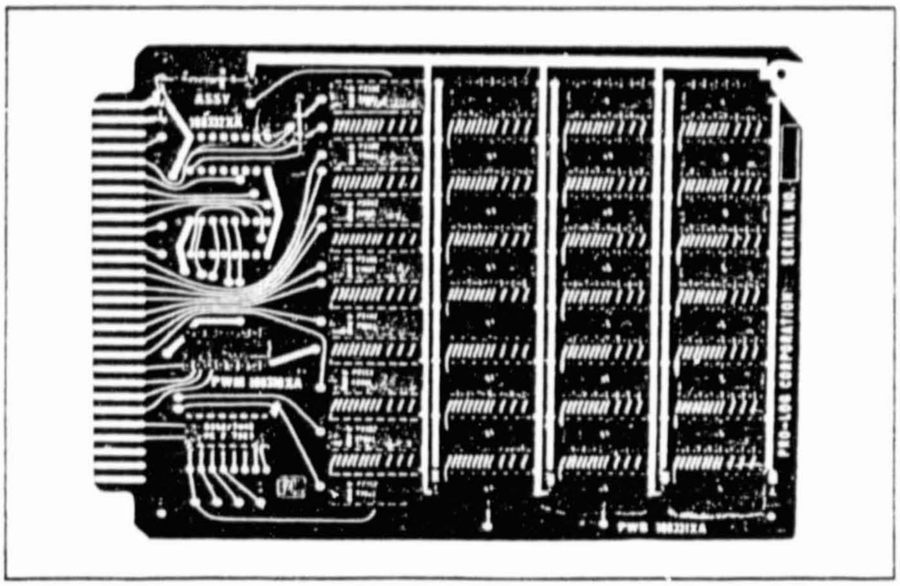
MPS SYSTEM COMPONENTS
8117 RAM
4K BYTES BY 8 BITS

A printed circuit card which implements 2102-4 static MOS RAM as read-write-memory for the MPS series of 8-bit microprocessors. The 8117 is organized as four 1024 byte increments and can be used as 1K, 2K, 3K, or 4K bytes of 8-bit memory.

FEATURES

- 4096 bytes of 8-bit RAM capacity
- Use as 1K, 2K, 3K, or 4K bytes of RAM
- Static MOS RAM
- Card addressing allows direct expansion to 64 K bytes
- 3-State data bus output





8117 RAM

CARD DIMENSIONS

- 4.50 in. (11.43 cm) high by 6.50 in. (16.51 cm) long
- 0.48 in. (1.22 cm) maximum profile thickness
- 0.062 in. (0.16 cm) printed circuit board thickness

CARD INCLUDES

- Card ejector
- 32 sockets
- Eight 2102-4 static RAM devices, 1024 bytes
- Chip select circuit
- Socket for card select option

INTERFACE

Inputs
(Active low logic, loading 1 TTL load, except where noted)

- 8 Data lines
- 10 Word Address, Active high, 32 MOS loads
- 3 Chip Address, Active high
- 1 Chip Enable, 2 TTL loads
- 3 Card Address, Active high
- 1 Card Enable
- 1 Memory Read
- 1 Memory Write, 32 MOS loads

Outputs (Active low, drive capability 10 TTL loads)

- 8 Data lines, 3-state
- 8 Card Select lines

EDGE CONNECTOR PIN LIST

Signal	Signal Flow	Pin Number	Pin Number	Signal Flow	Signal
+5 VOLTS	IN	2	1	IN	+5 VOLTS
GROUND	IN	4	3	IN	GROUND
		6	5		
DIN8*	OUT	8	7	IN	DOUT-8*
DIN7*	OUT	10	9	IN	DOUT-7*
DIN6*	OUT	12	11	IN	DOUT-6*
DIN5*	OUT	14	13	IN	DOUT-5*
DIN4*	OUT	16	15	IN	DOUT-4*
DIN3*	OUT	18	17	IN	DOUT-3*
DIN2*	OUT	20	19	IN	DOUT-2*
DIN1*	OUT	22	21	IN	DOUT-1*
CARD ADR-4	IN	24	23	IN	WORD ADR (A8)
CARD ADR-2 (A14)	IN	26	25	IN	WORD ADR (A7)
CARD ADR-1 (A13)	IN	28	27	IN	WORD ADR (A6)
RAM ADR-4	IN	30	29	IN	WORD ADR (A5)
RAM ADR-2 (A12)	IN	32	31	IN	WORD ADR (A4)
RAM ADR-1 (A11)	IN	34	33	IN	WORD ADR (A3)
WORD ADR (A10)	IN	36	35	IN	WORD ADR (A2)
WORD ADR (A9)	IN	38	37	IN	WORD ADR (A1)
CARD SEL-3*	OUT	40	39	OUT	CARD SEL-0*
CARD SEL-4*	OUT	42	41	IN	CARD ENABLE*
CARD SEL-5*	OUT	44	43	OUT	CARD SEL-1*
CARD SEL-6*	OUT	46	45	OUT	CARD SEL-2*
		48	47	IN	CHIP ENABLE*
		50	49	OUT	CARD SEL-7*
		52	51		
RDM*	IN	54	53		
WRM*	IN	56	55		

8117

POWER REQUIREMENTS

+VCC = +5 volts ±5% @ 1.6 Amps maximum (50 mA per RAM)
GND = 0 volts

OPERATING TEMPERATURE RANGE: 0-55°C

CONNECTOR REQUIREMENTS: 56 pin, 28 position dual-readout on 0.125 in. (0.318 cm) centers

APPLICATION NOTES

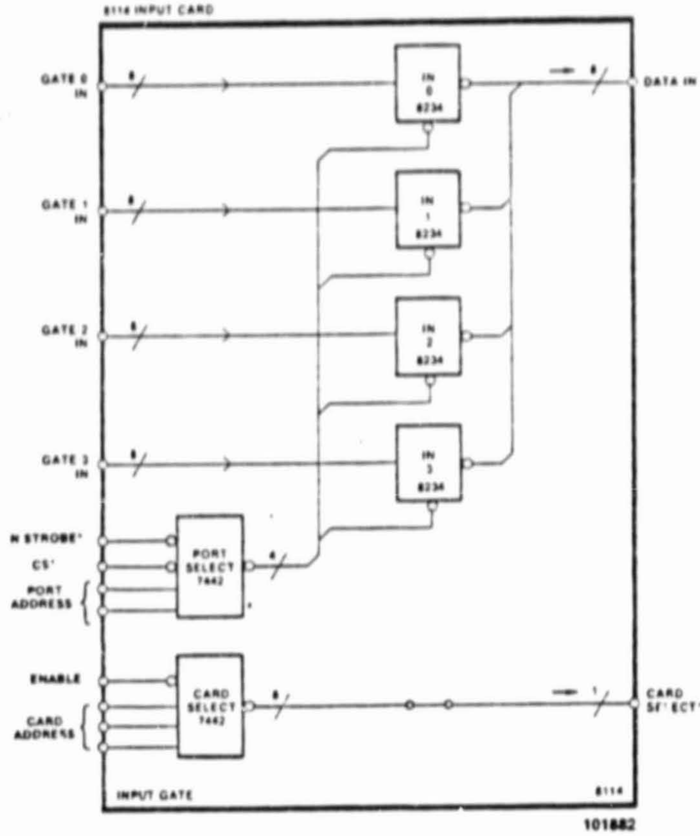
If only one memory card is used in the total system, tie the Chip Enable input to ground. If more than one memory card is used in the system, at least one Card Select circuit must be used and the Chip Enable lines individually wired to the Card Select line for the desired card address.

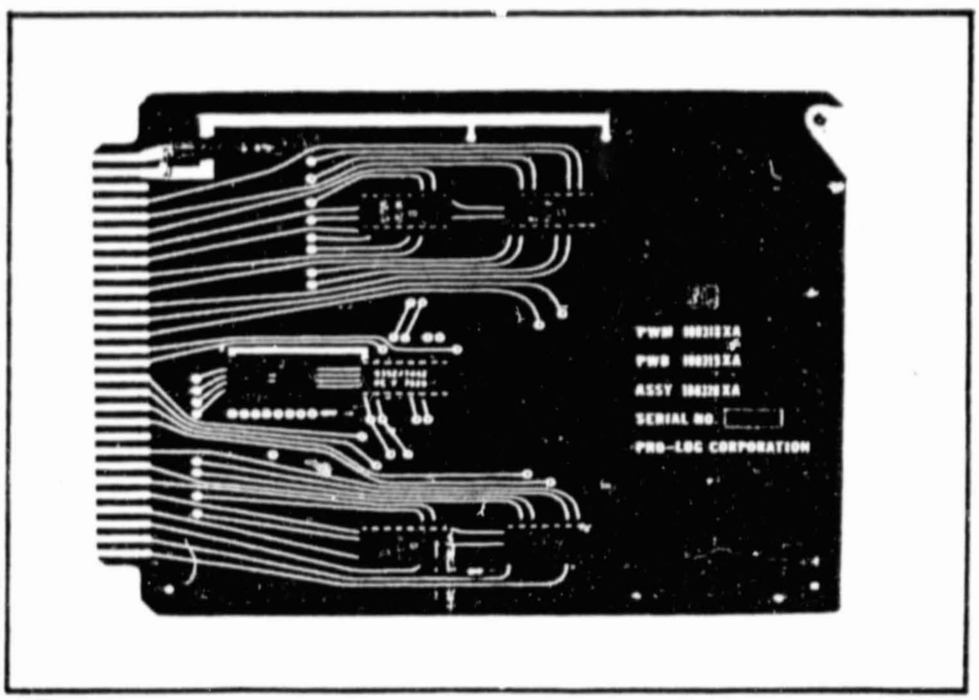
PRO-LOG
CORPORATION 2411 Garden Road Monterey, California 93940 Telephone (408) 372-4593
TWX: 910-360-7082

MPS SYSTEM COMPONENTS
8114 INPUT SELECTOR
8-BIT PORT

The 8114 provides four 8-bit selector gates as input ports for PRO-LOG's 8-bit microprocessors. The card includes addressing and gate control for transferring 8-bits of data from addressed gates to a data bus.

- Four 8-bit input selectors (32 lines)
- Port address decoding
- All integrated circuits socketed
- Card address decoding
- 3-state data in bus
- Input strobe





8114 INPUT

CARD DIMENSIONS

- 4.50 in. (11.43 cm) high by 6.50 in. (16.51 cm) long
- 0.48 in. (1.22 cm) maximum profile thickness
- 0.062 in. (0.16 cm) printed circuit board thickness

CARD INCLUDES

- Card Ejector
- Four dual 4-bit selector gates in sockets
- Port address logic
- Card select address decoder

INPUTS (Active low logic, loading 1 TTL load, except where noted)

- 32 Inputs, 4 groups of 8, Active high
- 2 Port address lines, Active high
- In strobe* (IN*)
- 3 Card address lines, Active high
- 1 Enable*
- 1 Card select (CS*)

OUTPUTS (Active low logic, drive capability 10 TTL loads)

- 8 Data lines, Open collector
- 1 Card Select*

8114 EDGE CONNECTOR PIN LIST

SIGNAL	SIGNAL FLOW	PIN NUMBER	PIN NUMBER	SIGNAL FLOW	SIGNAL
+5 VOLTS	IN	2	1	IN	+5 VOLTS
GND	IN	4	3	IN	GND
		6	5		
IN2-8	IN	8	7	IN	IN3-8
IN2-7	IN	10	9	IN	IN3-7
IN2-6	IN	12	11	IN	IN3-6
IN2-5	IN	14	13	IN	IN3-5
IN2-4	IN	16	15	IN	IN3-4
IN2-3	IN	18	17	IN	IN3-3
IN2-2	IN	20	19	IN	IN3-2
IN2-1	IN	22	21	IN	IN3-1
ENABLE* D	IN	24	23	OUT	DIN-8*
CARD ADDRESS C	IN	26	25	OUT	DIN-7*
CARD ADDRESS B	IN	28	27	OUT	DIN-6*
CARD ADDRESS A	IN	30	29	OUT	DIN-5*
PORT ADDRESS	IN	32	31	OUT	DIN-4*
PORT ADDRESS	IN	34	33	OUT	DIN-3*
CARD SELECT (CS*)	IN	36	35	OUT	DIN-2*
IN STROBE* (IN*)	IN	38	37	OUT	DIN 1*
CARD SELECT*	OUT	40	39		
IN1-8	IN	42	41	IN	IN0-8
IN1-7	IN	44	43	IN	IN0-7
IN1-6	IN	46	45	IN	IN0-6
IN1-5	IN	48	47	IN	IN0-5
IN1-4	IN	50	49	IN	IN0-4
IN1-3	IN	52	51	IN	IN0-3
IN1-2	IN	54	53	IN	IN0-2
IN1-1	IN	56	55	IN	IN0-1

* Designates active low logic

CONNECTOR REQUIREMENTS: 56 pin, 28 position dual-readout on 0.125 in. (0.328 cm) centers

POWER REQUIREMENTS

+VCC = +5 volts +5% @ 200 mA maximum
GND = 0 volts

OPERATING TEMPERATURE RANGE: 0-55°C

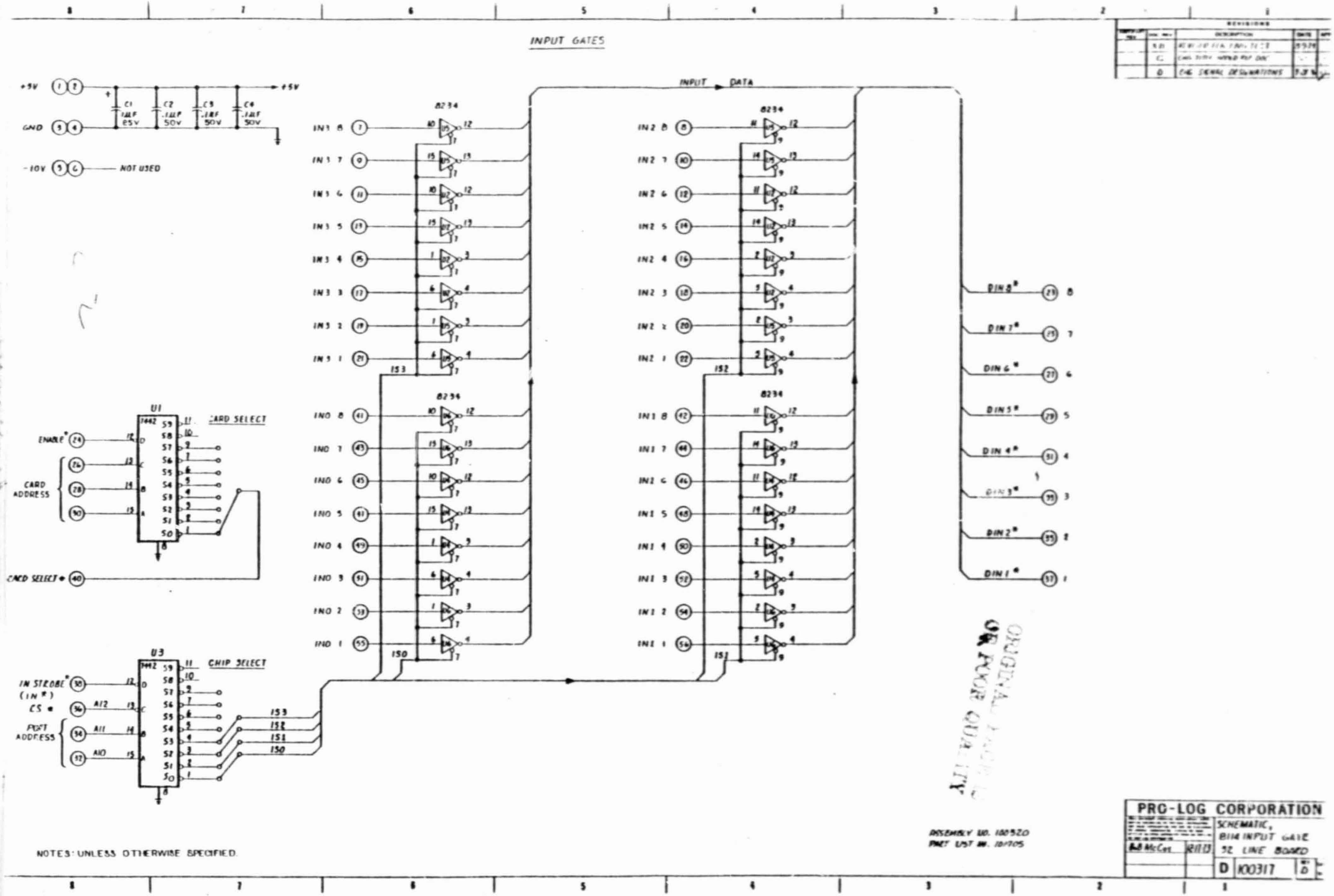
APPLICATION NOTE

If only one input card is used in the total system, tie the IN STROBE* (IN*) Pin 38 to the processor input strobe signal (IN*) and ground CS* Pin 36. If more than one input card is used in the system, connect the card select* line Pin 40 to CS* Pin 36 and the enable* line to ground. The card select circuit must be hard wire jumpered on the card for the correct card address. See Application Note 109 for further information.

PRO-LOG CORPORATION

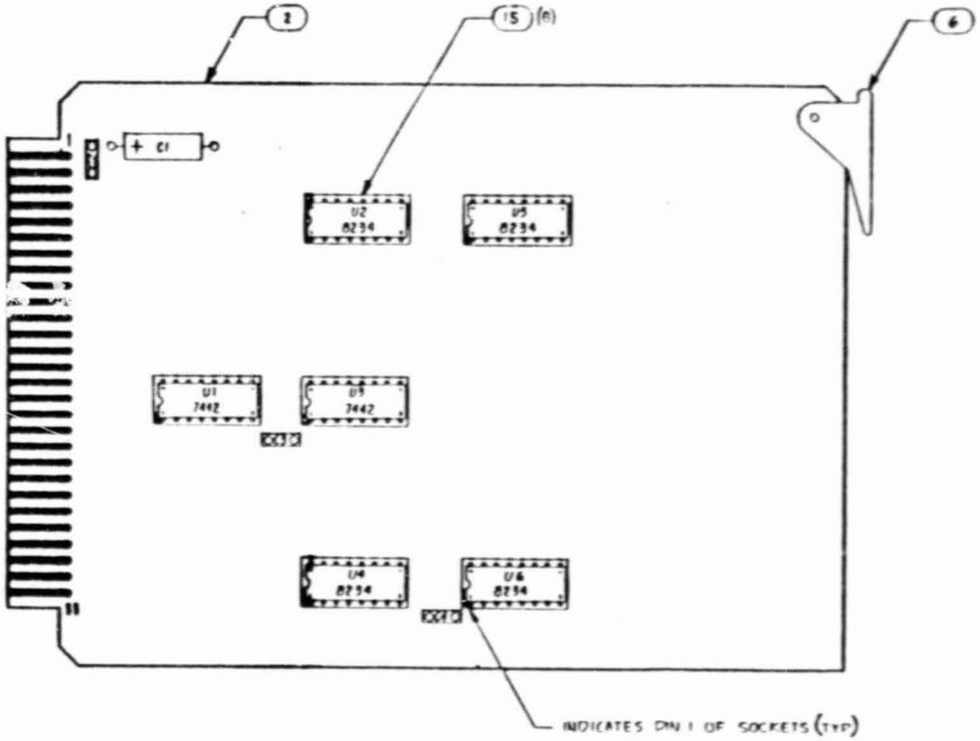
TWX 910-360-7082

2411 Garden Road Monterey, California 93940 Telephone (408) 372-4593



Wiring schematic for 8114 input card

REV	DESCRIPTION
B	DWG TITLE, IDENTIFICATION SOCKETS NOTE 1
C	ADDED ITEM 15



NOTES: UNLESS OTHERWISE SPECIFIED.

1. FOR ASSEMBLY PROCEDURES SEE 051004

SCHEMATIC NO. 100317
PARTS LIST NO. 101705

ITEM	DESCRIPTION	REF DESIGNATION
9	1 µF 50V	C2, C3, C4
8	1 µF 25V	C1

PRO-LOG CORPORATION

ASSY, 8114 INPUT GATE 32 LINE BOARD

BOB McCOY 12/20/78

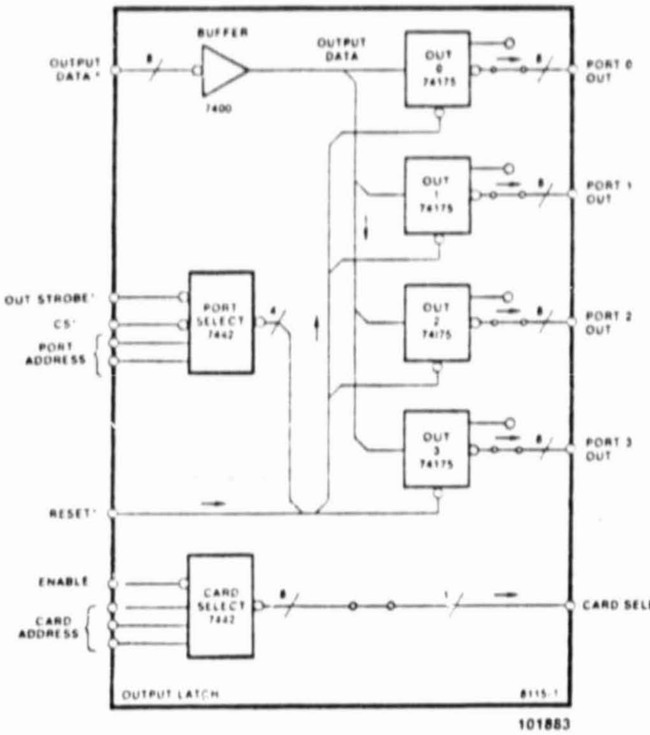
D 100320 C

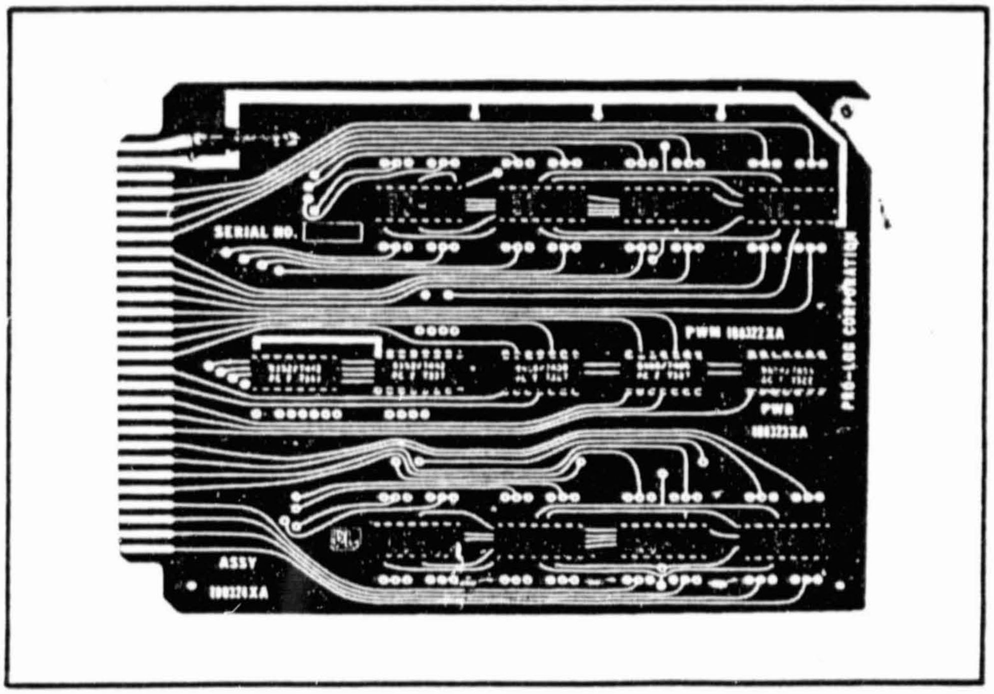
MPS SYSTEM COMPONENTS
8115-1 OUTPUT LATCH
8 BIT PORT

The 8115-1 card provides four 8-bit TTL latches as output ports for use with PRO-LOG's 8-bit microprocessor systems. The card includes address decode and strobe control for transferring 8-bits of data from a data bus to one of four addressed output latches.

FEATURES

- Four 8-bit latches (32 lines)
- Card address decoding
- All integrated circuits socketed
- Port address decoding
- Reset input
- Data bus buffered





8115-1 OUTPUT CARD

8115-1 OUTPUT LATCH CARD — SPECIFICATIONS

CARD DIMENSIONS

- 4.50 in. (11.43 cm) high by 6.50 in. (16.51 cm) long
- 0.48 in. (1.22 cm) maximum profile thickness
- 0.062 in. (0.16 cm) printed circuit board thickness

CARD INCLUDES

- Card ejector
- Eight 4-bit latches in sockets (8-bit ports)
- Port select logic
- Card select address decoder

INPUTS (Active low logic, loading 1 TTL load, except where noted)

- Enable*
- 2 Port address lines, Active high
- 1 Card select (CS*)
- 3 Card address lines, Active high
- 1 Reset
- 1 Out strobe* (OUT*)

OUTPUTS (Active low logic. Drive capability 10 TTL loads, except where noted)

- 32 Outputs, 4 groups of 8; Selectable, active low or high
- 1 Card select*

8115-1 EDGE CONNECTOR PIN LIST

Signal	Signal Flow	Pin Number	Pin Number	Signal Flow	Signal
+5 VOLTS	IN	2	1	IN	+5 VOLTS
GROUND	IN	4	3	IN	GROUND
		6	5		
OUT 2 - 8	OUT	8	7	OUT	OUT 3 - 8
OUT 2 - 7	OUT	10	9	OUT	OUT 3 - 7
OUT 2 - 6	OUT	12	11	OUT	OUT 3 - 6
OUT 2 - 5	OUT	14	13	OUT	OUT 3 - 5
OUT 2 - 4	OUT	16	15	OUT	OUT 3 - 4
OUT 2 - 3	OUT	18	17	OUT	OUT 3 - 3
OUT 2 - 2	OUT	20	19	OUT	OUT 3 - 2
OUT 2 - 1	OUT	22	21	OUT	OUT 3 - 1
ENABLE* D	IN	24	23	IN	DOUT 8*
CARD ADDRESS C	IN	26	25	IN	DOUT 7*
CARD ADDRESS B	IN	28	27	IN	DOUT 6*
CARD ADDRESS A	IN	30	29	IN	DOUT 5*
PORT ADDRESS	IN	32	31	IN	DOUT 4*
PORT ADDRESS	IN	34	33	IN	DOUT 3*
CARD SELECT (CS*)	IN	36	35	IN	DOUT 2*
OUT STROBE* (OUT*)	IN	38	37	IN	DOUT 1*
CARD SELECT*	OUT	40	39	IN	RST*
OUT 0 - 8	OUT	42	41	OUT	OUT 1 - 8
OUT 0 - 7	OUT	44	43	OUT	OUT 1 - 7
OUT 0 - 6	OUT	46	45	OUT	OUT 1 - 6
OUT 0 - 5	OUT	48	47	OUT	OUT 1 - 5
OUT 0 - 4	OUT	50	49	OUT	OUT 1 - 4
OUT 0 - 3	OUT	52	51	OUT	OUT 1 - 3
OUT 0 - 2	OUT	54	53	OUT	OUT 1 - 2
OUT 0 - 1	OUT	56	55	OUT	OUT 1 - 1

* Designates active low logic.

POWER REQUIREMENTS

- +VCC = +5 volts ±5% at 500 mA maximum
- GND = 0 volts

OPERATING TEMPERATURE RANGE: 0-55°C

CONNECTOR REQUIREMENTS: 56 pin, 28 position dual-readout on 0.125 in. (0.318 cm) centers

Wired for two or more cards

APPLICATION NOTE

If only one card is used ground CS* Pin 36 and wire the out* enable to the output strobe Pin 38. If more than one output card is used in the system, connect the card select* line Pin 40 to CS* Pin 36 and the enable* line Pin 24 to GND. The card select decoder must be hard wire jumpered on the card for the correct card address. See Application Note 109 for further information.

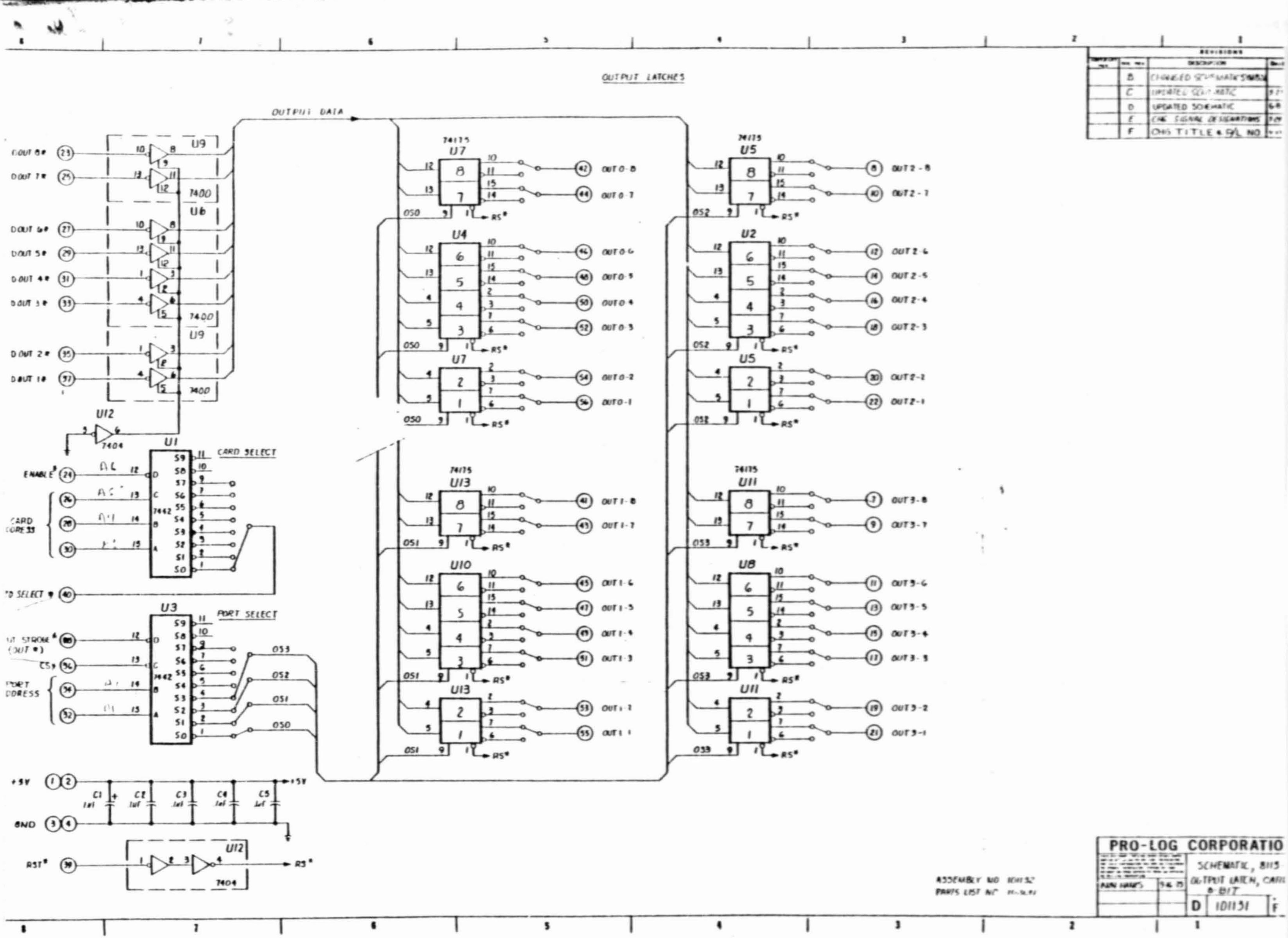
Individual output lines are connected for active high logic. The output lines can optionally be connected for active low logic by jumpers on the card.

101583 9/76

PRO-LOG CORPORATION

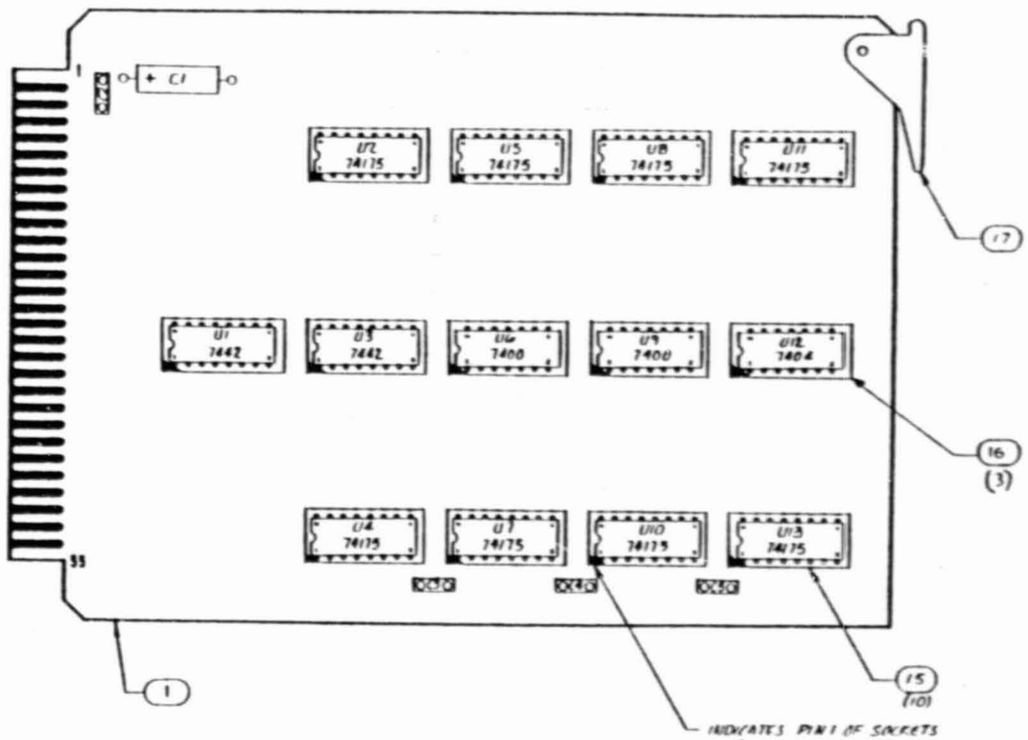
TWX: 910-360-7082

2411 Garden Road Monterey, California 93940 Telephone (408) 372-4593



Wiring schematic for 8115-1 output card

	A	CH. ITEM 15 & PL NO.	



ORIGINAL PAGE IS OF POOR QUALITY

FOR ASSEMBLY PROCEDURES SEE AS1004

NOTES: UNLESS OTHERWISE SPECIFIED.

SCHEMATIC NO. 101131
PART LIST NO. 102891

ITEM	DESCRIPTION	REF DESIGNATION
8	.1 µF 50V	C2-C5
7	1 µF 25V	C1

PRO-LOG CORPORATION

ASSEMBLY, BUS-I OUTPUT LATCH, 8 BIT

D 101132 A

8407 SERIAL INTERFACE FOR TTY AND RS232-C

A printed circuit card which provides the electrical interface between a microprocessor and both RS-232 and TTY serial data communications lines.

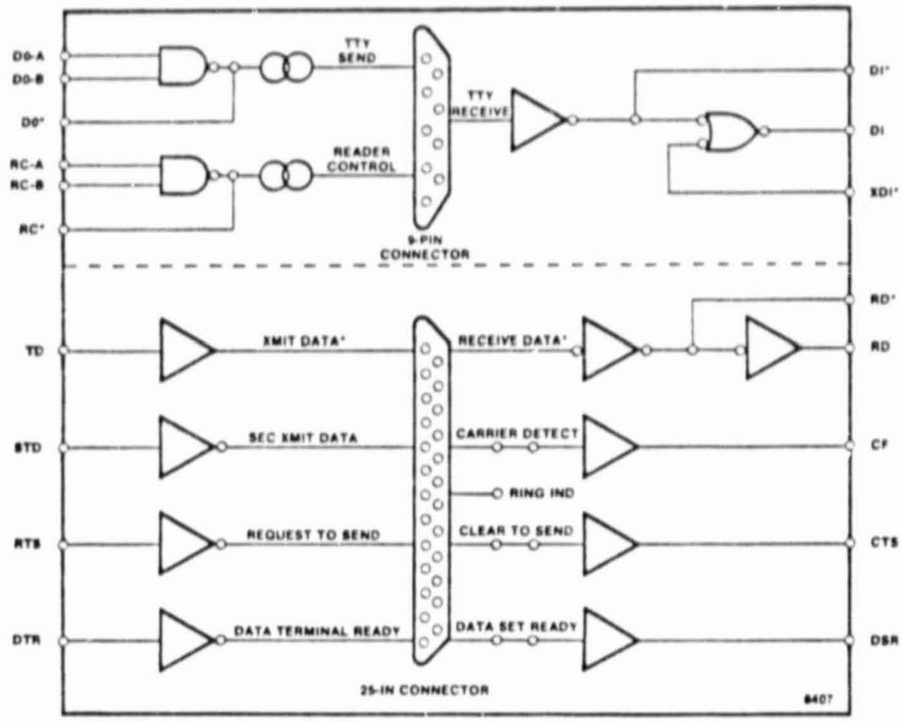
The RS-232 portion implements EIA Standard RS-232-C drivers, receivers, and D-type connector configured to place the 8407 in the Data Terminal position.

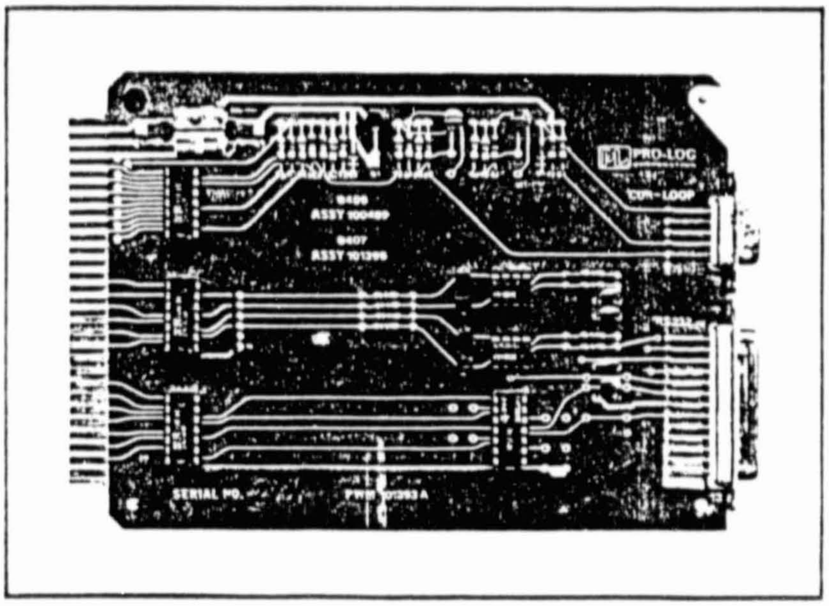
The TTY portion consists of two 20 milliamp current loops for full duplex send and receive capability, with a relay driver for remote control of ASR-33 console tape readers.

The 8407 has card-front interface cable connectors with a TTL interface to microprocessor I/O ports at the card edge connector. The microprocessor used in conjunction with the 8407 must provide timing and serial-to-parallel conversion for both interface circuits.

FEATURES

- **Pluggable card-front D-type interface connectors**
- **TTY full duplex 20 mA current loop**
- **Remote TTY reader relay driver**
- **RS-232-C interface meets EIA standard for data terminal**





8407 SERIAL INTERFACE

8407 SERIAL INTERFACE

SPECIFICATIONS

CARD DIMENSIONS

- 11.43 cm (4.50 in) high by 16.51 cm (6.50 in) long
- 1.27 cm (0.50 in) maximum profile thickness
- 0.160 cm (0.062 in) PWB thickness

CARD INCLUDES

- Card ejector
- Sockets for all ICs
- Card front interface cable connectors

RS-232 CIRCUIT INCLUDES

- EIA Standard 25-pin D-type cable connector
- Connector outputs:
  - Transmitted Data (circuit BA)
  - Secondary Transmitted Data (circuit SBA)
  - Request To Send (circuit CA)
  - Data Terminal Ready (circuit CD)
  - Signal Ground (circuit AB)
- Connector inputs:
  - Receive Data (circuit BB)
  - Carrier Detect (circuit CF)
  - Ring Indicator (circuit CE)
  - Data Set Ready (circuit CC)
  - Clear To Send (circuit CB)

TTY CIRCUIT INCLUDES

- 9-pin D-type cable connector
- Connector signals:
  20 mA Send to TTY
  20 mA Receive from TTY
  15V, 200 ohm source for tape reader control relay drive
- Spare gating for TTY selection by Processor

POWER REQUIREMENTS

+VCC = +5V ± 10% @ 140 mA
GND = 0V
-VDD = -9 to -12V @ 120 mA

OPERATING TEMPERATURE RANGE: 0-55°C

CONNECTOR REQUIREMENTS

- 56 pin, 28 position dual readout
- 0.318 cm (0.125 in) centers

8407 EDGE CONNECTOR PIN LIST

SIGNAL	SIGNAL FLOW	PIN NUMBER	PIN NUMBER	SIGNAL FLOW	SIGNAL
+5 VOLTS	IN	2	1	IN	+5 VOLTS
GROUND	IN	4	3	IN	GROUND
-9 TO -12 VOLTS	IN	6	5	IN	-5 TO -12 VOLTS
PULL-UP	OUT	8	7	IN	D0*
RC*	IN	10	9	IN	D0-A
RC-B	IN	12	11	IN	D0-B
RC-A	IN	14	13	OUT	DI
NAND OUT*(SPARE)	OUT	16	15	IN	KDI*
NAND IN-A (SPARE)	IN	18	17	OUT	DI*
NAND IN-B (SPARE)	IN	20	19		
		22	21		
		24	23		
		26	25	OUT	INVERTER*(SPARE)
		28	27	IN	INVERTER (SPARE)
		30	29	IN	XMIT DATA*
		32	31	IN	REQ TO SEND
		34	33	IN	DATA TERM RDY
		36	35	IN	SUP XMIT DATA
		38	37		
		40	39		
		42	41	OUT	INVERTER*(SPARE)
		44	43	IN	INVERTER(SPARE)
		46	45	OUT	CARRIER DETECT
		48	47	OUT	DATA SET READY
		50	49	OUT	CLEAR TO SEND
		52	51	OUT	RECEIVE DATA
		54	53	OUT	RECEIVE DATA*
		56	55		

9 PIN "D" CONNECTOR PIN LIST

SIGNAL	SIGNAL FLOW	PIN NUMBER	PIN NUMBER	SIGNAL FLOW	SIGNAL
		2	1	OUT	-RELAY (-VR)
-DATA FROM TTY	IN	4	3		
+RELAY (+VR)	OUT	6	5	IN	+INTERLOCK
-DATA TO TTY	OUT	8	7	OUT	+DATA TO TTY
			9	IN	+DATA FROM TTY

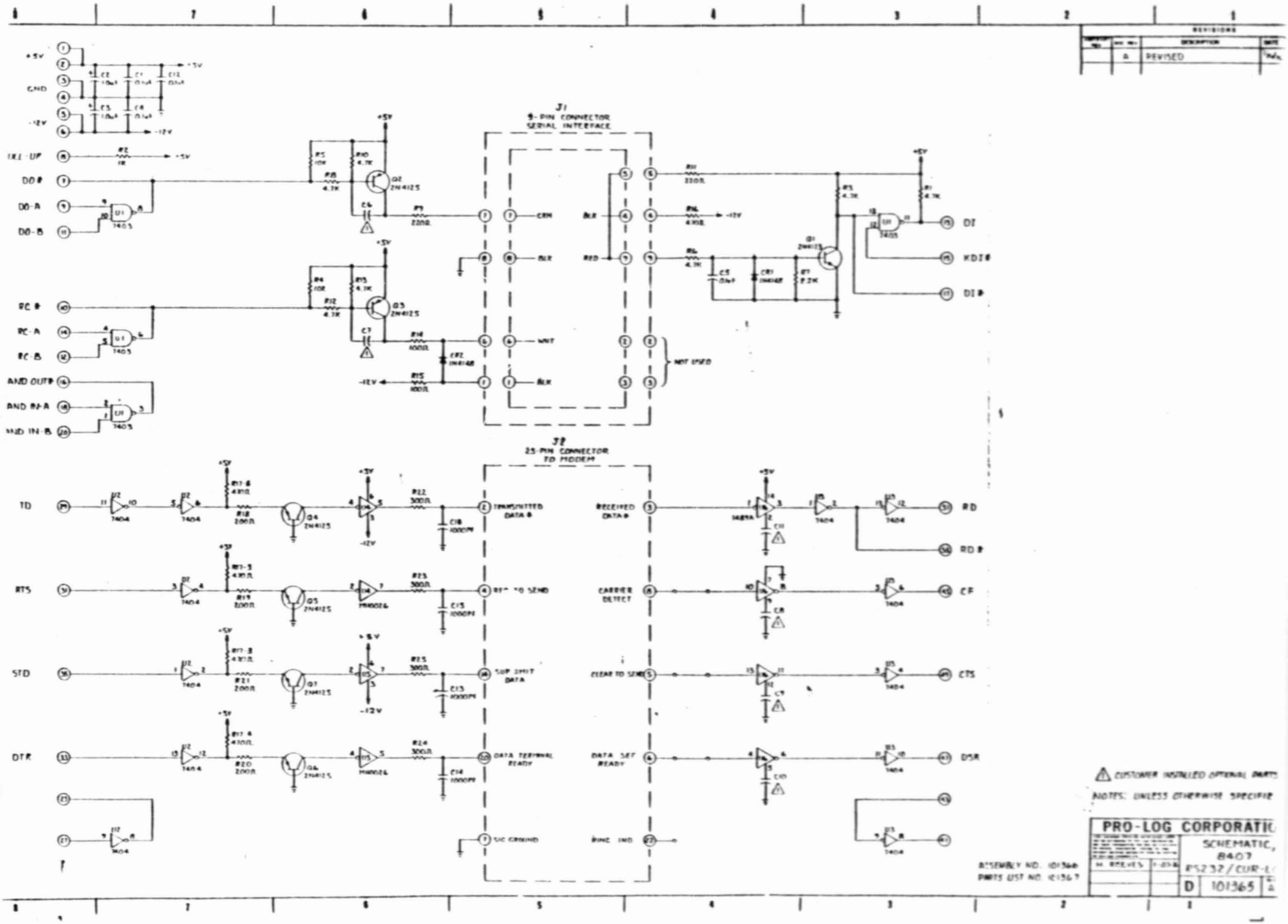
25 PIN "D" CONNECTOR PIN LIST

SIGNAL	SIGNAL FLOW	PIN NUMBER	PIN NUMBER	SIGNAL FLOW	SIGNAL
TRANSMIT DATA*	OUT	2	1		
REQ TO SEND	OUT	4	3	IN	RECEIVED DATA*
DATA SET READY	IN	6	5	IN	CLEAR TO SEND
CARRIER DET	IN	8	7	IN	SIGNAL GND
		10	9		
		12	11		
SUP XMIT DATA	OUT	14	13		
		16	15		
		18	17		
DATA TERM READY	OUT	20	19		
RING IND	IN	22	21		
		24	23		
			25		

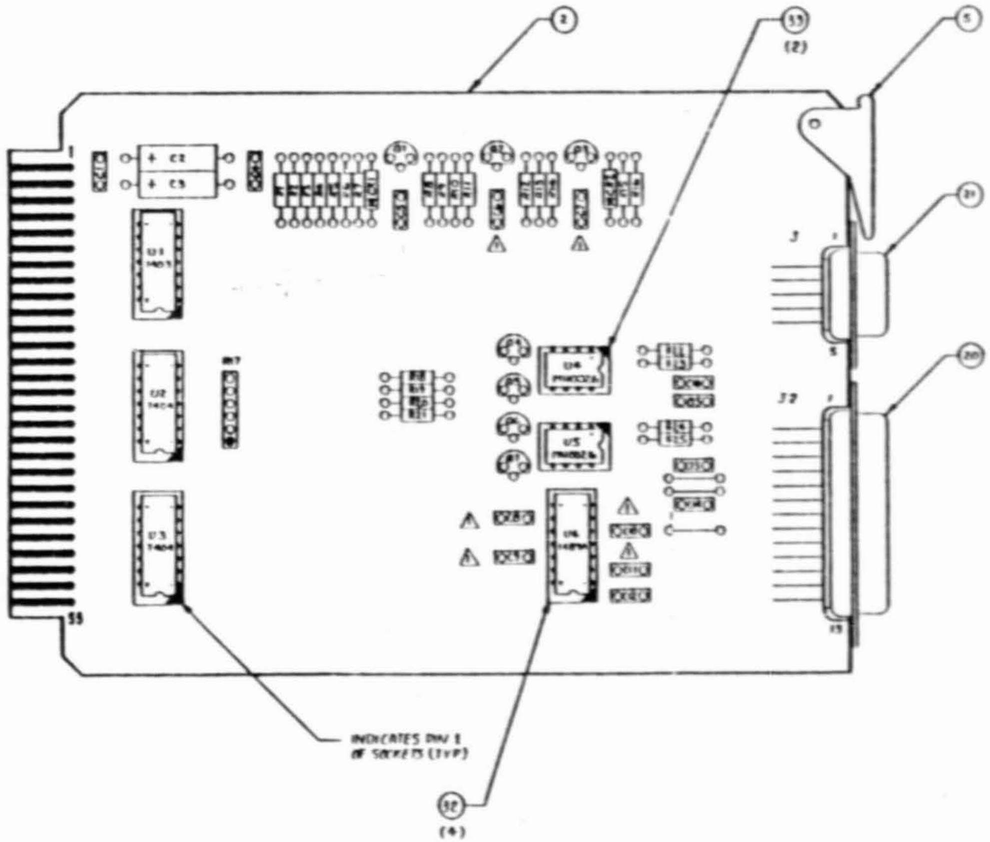
*Designates Active Low Level Logic

PRO-LOG
CORPORATION 2411 Garden Road Monterey, California 93940 Telephone (408) 372-4593
TWX: 910-360-7082

Wiring schematic for 8407 serial interface card



REV	DESCRIPTION	DATE
A	ADDED SOCKETS	



ORIGINAL PAGE IS OF POOR QUALITY

NOTES: UNLESS OTHERWISE SPECIFIED.

△ CUSTOMER INSTALLED OPTIONAL PARTS.

2. FOR ASSEMBLY PROCEDURES SEE AS1004.

SCHEMATIC NO. 101365
PARTS LIST NO. 101367

ITEM	DESCRIPTION	REF DESIGNATION
30	470Ω NETWORK	R17
29	10K CC 1/4W	R4,R5
28	4.7K	[illegible]
27	1K	R2,R7
26	470Ω	R16
25	300Ω	[illegible]
24	220Ω	[illegible]
23	1000Ω	R18-21
22	100Ω CC 1/4W	[illegible]
16	1N4148 DO-35	CR1,CR2
15	2N4123 TO-92	[illegible]
14	2N4125 TO-92	[illegible]
13		C6-C11
12	1000PF	C15-C16
11	0.1uF 50V	[illegible]
10	1.0uF 25V	[illegible]

PRO-LOG CORPORATION

ASSEMBLY, 8407 RS232/CUR-LOOP

W. REEVES

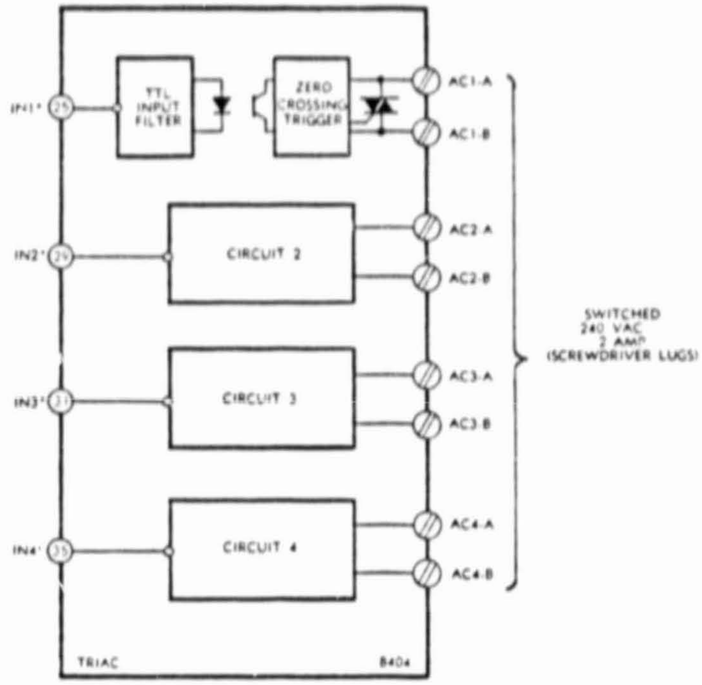
D 101366 A

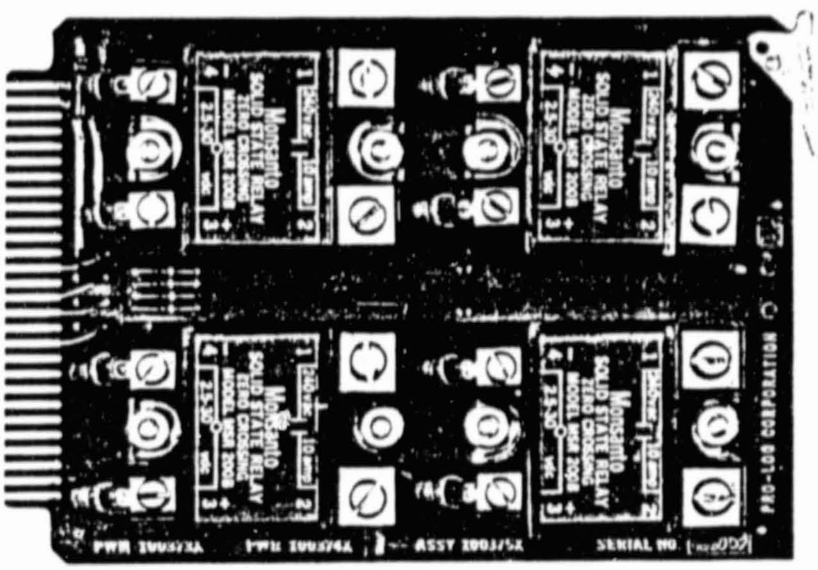
MICROPROCESSOR INTERFACE CARDS
8404 TRIAC - 4

A printed circuit card which implements up to four solid state power relays (TRIACS) for use with PLS-400 or MPS-800 microprocessor systems. The 8404 provides buffered TTL control of up to four 240 VAC 2 Amp isolated circuits.

FEATURES

- Up to four normally open 240 VAC Triacs
- Screwdriver lug cable attachment





8404 TRIAC

CARD DIMENSIONS (Requires two card rack slots on 0.50" spacing)

- 4.50 in. (11.43 cm) high by 6.50 in. (16.51 cm) long
- 0.48 in. (1.22 cm) maximum profile thickness
- 0.062 in. (0.16 cm) printed circuit board thickness

CARD INCLUDES

- Two 240 VAC 10 Amp triacs 8404-2
- Four 240 VAC 10 Amp triacs 8404-4

AC OUTPUT INTERFACE

- Screwdriver lug cable attachment
- Lug requirement: #8 minimum or bare wire
- 240 VAC, 2 Amp max (Free air mounting, TA = 85° C)

PC EDGE CONNECTOR INTERFACE

- 56 pin, 28 position, dual-readout on 0.125 centers.
- 4 Control Inputs, active low logic, 5 TTL loads

POWER REQUIREMENTS

+VCC = +5 volts ±5% @ 30 milliamp maximum
GND = 0 volts

OPERATING TEMPERATURE RANGE: 0-55°C

EDGE CONNECTOR PIN LIST

SIGNAL	SIGNAL FLOW	PIN NUMBER	PIN NUMBER	SIGNAL FLOW	SIGNAL
+5 VOLTS	IN	2	1	IN	+5 VOLTS
GROUND	IN	4	3	IN	GROUND
		6	5		
		8	7		
		10	9		
		12	11		
		14	13		
		16	15		
		18	17		
		20	19		
		22	21		
		24	23		
		26	25	IN	TTL IN1*
		28	27		
		30	29	IN	TTL IN2*
		32	31	IN	TTL IN3*
		34	33		
		36	35	IN	TTL IN4*
		38	37		
		40	39		
		42	41		
		44	43		
		46	45		
		48	47		
		50	49		
		52	51		
		54	53		
		56	55		

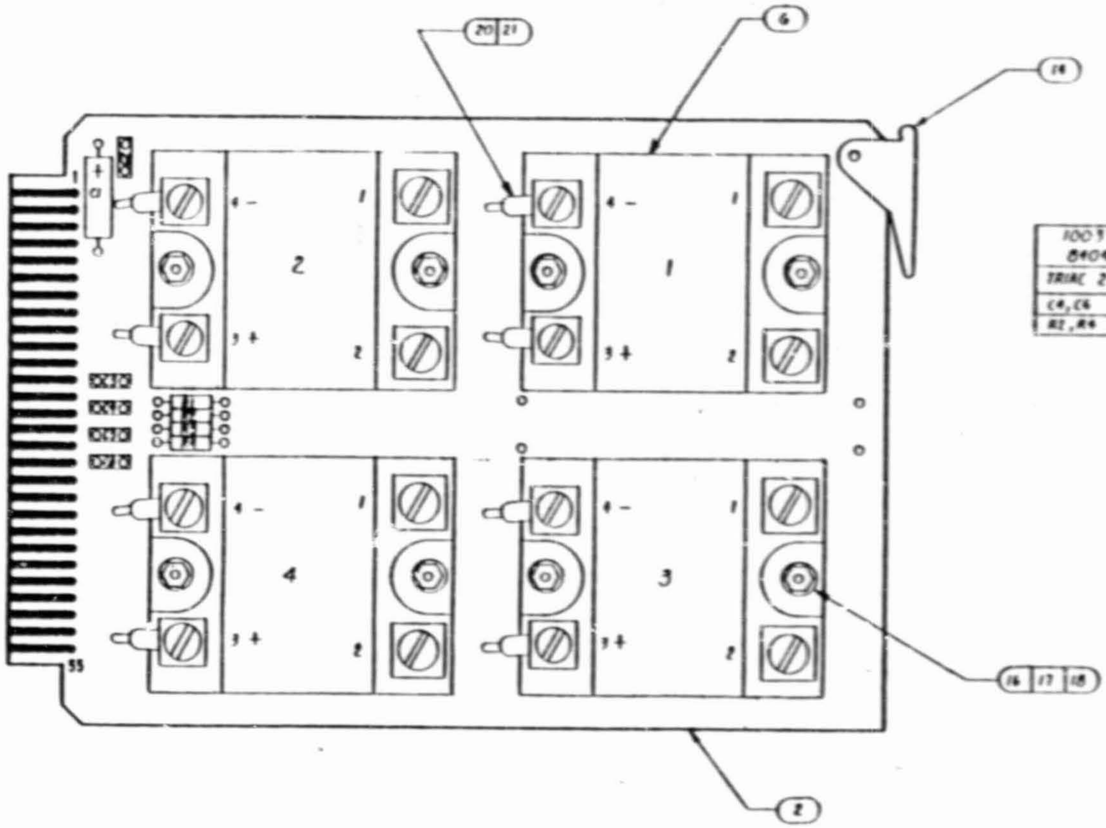
APPLICATION NOTES

Although the TRIACS on the 8404 card are rated at 240 VAC and 10 Amps, the power at full rating cannot be dissipated on the printed circuit card. It is recommended where full rated operation is desired that the triacs be mounted on a heat sink and interfaced using the 8401 or 8405 interface cards.

When high voltage and high currents are being switched, noise transients can be generated which are disruptive to logic systems. It is recommended that when triacs must switch high voltage or currents that they be mounted remote from the logic system and as close as possible to the equipment being switched.

PRO-LOG
CORPORATION 2411 Garden Road Monterey, California 93940 Telephone (408) 372-4593
TWX: 910-360-7082

REV	DESCRIPTION
B	CHG TITLE, ADDED ASSY NO & P/L NO.



100375 8404-2	100683 8404-4
TRIAC 2 & 4	TRIAC 1,2,3,4
C4, C6	C3, C4, C5, C6
R2, R4	R1, R2, R3, R4

ITEM	DESCRIPTION	REF DESIGNATION
11	1K CC 1/4W	R1, R2, R3, R4
10	.01UF 50V	C3, C4, C5, C6
9	.1UF 50V	C2
8	1UF 25V	C1

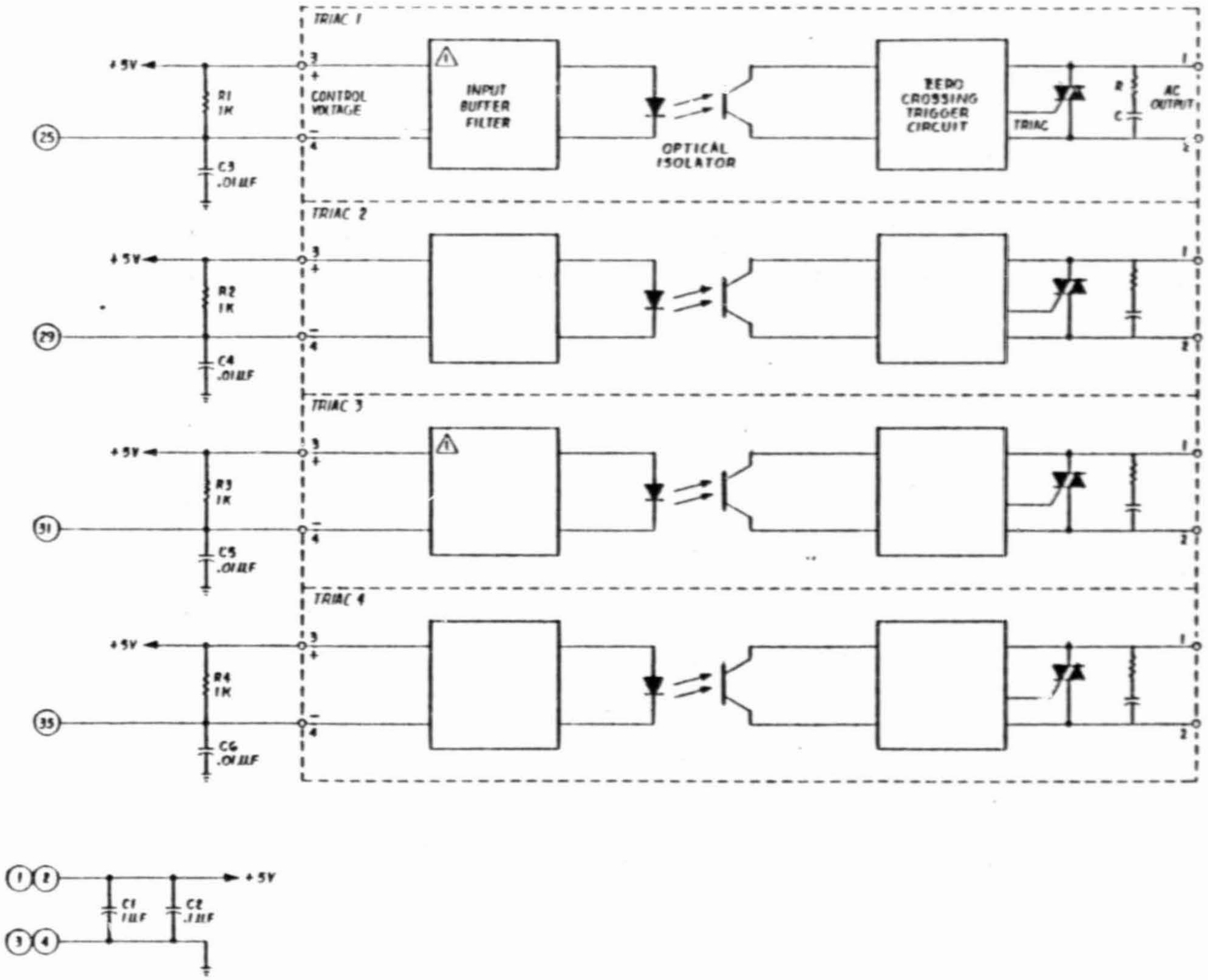
ASSEMBLY NO 100376
PARTS LIST NO 104133

PRO-LOG CORPORATION

ASSEMBLY 8404 TRIAC OUTPUT CARD

D 100375 B

Wiring schematic for 8404-4 triac card



THESE TRIACS OMITTED ON 8104-2.
NOTES UNLESS OTHERWISE SPECIFIED.

SCHEMATIC NO 100376
PARTS LIST NO 104138

PRO-LOG CORPORATIO

SCHEMATIC 8404-
TRIAC OUTPUT CA

BOB McCOY

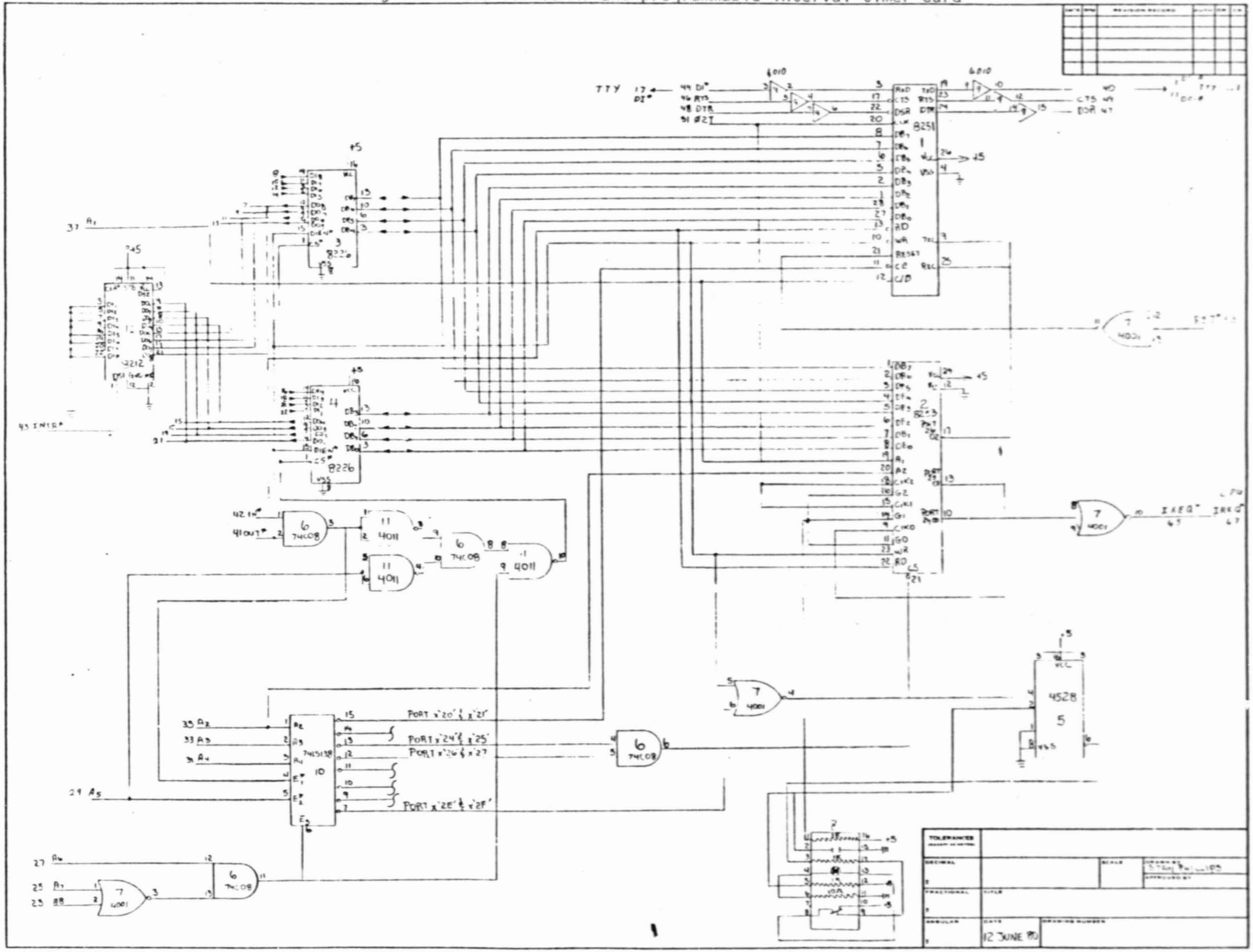
D 100376

107

BACKPLANE WIRING FOR 8404 TRIAC CARD

Signal Name		Wirewrap Pins	
From	To	From	To
+5V	+5V	J10-1	J17-1
+5V	+5V	J10-2	J17-2
Ground	Ground	J10-3	J17-3
Ground	Ground	J10-4	J17-4
OUT 1-1	TTL IN1*	J10-55	J17-25
OUT 1-2	TTL IN2*	J10-53	J17-29
OUT 1-3	TTL IN3*	J10-51	J17-31
OUT 1-4	TTL IN4*	J10-49	J17-35

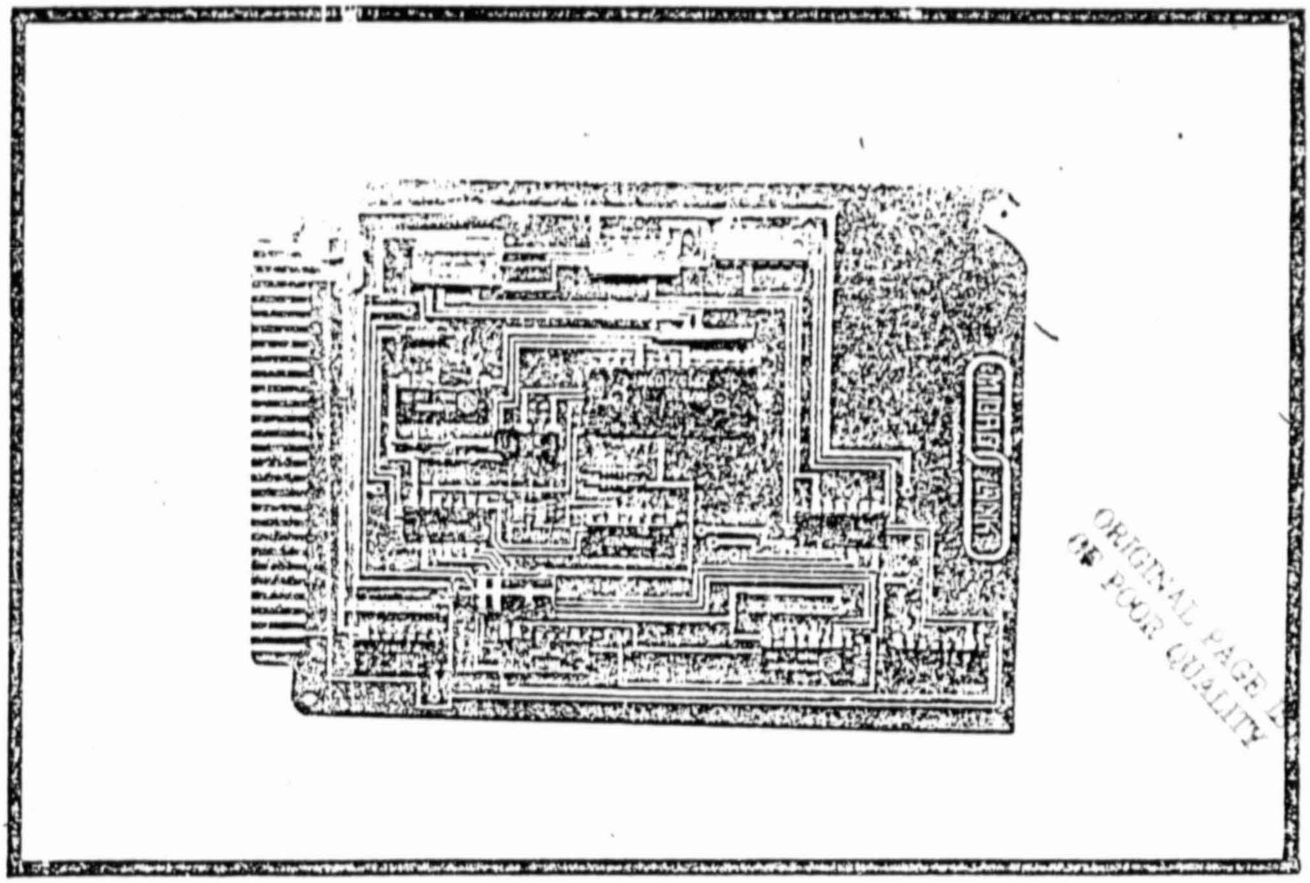
Wiring schematic for USART and programmable interval timer card

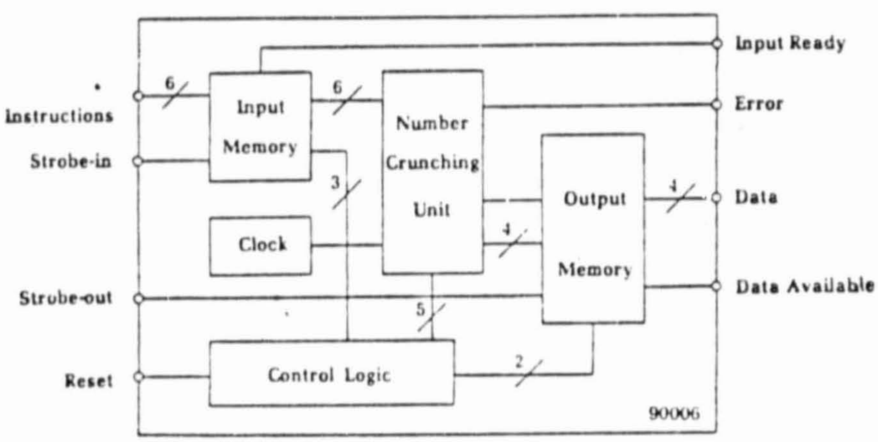


The Calculator Card is a Micro Systems Component Card which provides number crunching abilities to microprocessor based systems without the need for extensive software.

FEATURES:

- Parallel processing of up to 64 instructions.
- Inputs buffered to 1 TTL load.
- Output fan out of 10 TTL loads.
- Key level instruction language (RPN).
- +, -, x, ÷, 1/x, $\sqrt{x}$, x^2, 10^x, log x, e^x, lnx, Y^x, Trigonometric and Inverse Trigonometric Functions, π, Degrees ↔ Radians.
- Four register stack and one memory register.
- Floating point or exponential notation.





Specifications

CARD DIMENSIONS
- 4.5 in (11.43 cm) high by 6.5 in (16.51 cm) long
- 0.5 in (1.28 cm) maximum profile thickness
- 0.062 in (0.16 cm) printed circuit board thickness

CARD INCLUDES:
- Card ejector
- 64 byte input and output memory buffers
- Instruction STROBE-IN and data STROBE-OUT lines
- Calculation ERROR flag
- Input memory READY flag
- DATA AVAILABLE flag
- Master RESET
- Four DATA lines

INPUTS:
- 6 Instruction lines - Parallel in (active high)
- Instruction STROBE-IN line (active high)
- Data STROBE-OUT line (active high)
- RESET (active low)

OUTPUTS:
- Input READY (active high)
- DATA AVAILABLE (active high)
- Calculation ERROR (active low)
- BCD Data (active high)

POWER REQUIREMENTS:
- VCC = +5 volts, ±5% 0.20 amps at 25°C
- VDD = -10 volts, ±5% 0.18 amps at 25°C
- GND = 0.0 volts

CONNECTOR REQUIREMENTS:
- 56 pin, 28 position dual-readout on 0.125 in (0.318 cm) centers

CALCULATOR CARD 90006

EDGE CONNECTOR PIN LIST					
SIGNAL	SIGNAL FLOW	PIN NUMBER	PIN NUMBER	SIGNAL FLOW	SIGNAL
+5 VOLTS	IN	2	1	IN	+5 VOLTS
GROUND	IN	4	3	IN	GROUND
-10 VOLTS	IN	6	5	IN	-10 VOLTS
		8	7		
		10	9		
STROBE-IN	IN	12	11		
INST. 0	IN	14	13		
INST. 1	IN	16	15		
INST. 2	IN	18	17		
INST. 3	IN	20	19		
INST. 5	IN	22	21		
INST. 4	IN	24	23		
RESET •	IN	26	25		
		28	27		
		30	29		
		32	31		
		34	33		
		36	35		
		38	37		
DATA AVAILABLE •	OUT	40	39		
ERROR •	OUT	42	41		
DATA 2	OUT	44	43		
DATA 4	OUT	46	45		
DATA 8	OUT	48	47		
DATA 1	OUT	50	49		
STROBE-OUT	IN	52	51		
		54	53	OUT	READY
		56	55		

• DESIGNATES ACTIVE LOW LEVEL LOGIC

REPRESENTED BY:

MICRO LINK

BACKPLANE WIRING FOR CALCULATOR CARD

Signal Name		Wirewrap Pins	
From	To	From	To
IN 3-8	READY	J8-7	J12-53
IN 3-7	DATA AVAILABLE*	J8-9	J12-40
IN 3-6	ERROR*	J8-11	J12-42
IN 3-4	DATA 8	J8-15	J12-48
IN 3-3	DATA 4	J8-17	J12-46
IN 3-2	DATA 2	J8-19	J12-44
IN 3-1	DATA 1	J8-21	J12-50
OUT 3-8	STROBE-OUT	J10-7	J12-52
OUT 3-7	STROBE-IN	J10-9	J12-12
OUT 3-6	INSTR-5	J10-11	J12-22
OUT 3-5	INSTR-4	J10-13	J12-24
OUT 3-4	INSTR-3	J10-15	J12-20
OUT 3-3	INSTR-2	J10-17	J12-18
OUT 3-2	INSTR-1	J10-19	J12-16
OUT 3-1	INSTR-0	J10-21	J12-14
OUT 2-8	RESET *	J10-8	J12-26

BACKPLANE WIRING TO MOISTURE METER

Signal Name		Wirewrap Pins	
From	To	From	To
Ground	Ground	J8-3	Ground
Ground	Ground	J8-4	Ground
INO-1	.1	J8-55	1
INO-2	.2	J8-53	2
INO-3	.4	J8-51	3
INO-4	.8	J8-49	4
INO-5	1	J8-47	5
INO-6	2	J8-45	6
INO-7	4	J8-43	7
INO-8	8	J8-41	8
INI-1	10	J8-56	9
INI-2	20	J8-54	10
INI-3	40	J8-52	11
INI-4	80	J8-50	12

BACKPLANE WIRING FOR USART AND PROGRAMMABLE
INTERVAL TIMER CARD

Signal Name	Wirewrap Pins From	Wirewrap Pins To
DOUT 8*	J3-7	J16-7
DOUT 7*	J3-9	J16-9
DOUT 6*	J3-11	J16-11
DOUT 5*	J3-13	J16-13
DOUT 4*	J3-15	J16-15
DOUT 3*	J3-17	JL6-17
DOUT 2*	J3-19	J16-19
DOUT 1*	J3-21	J16-21
DIN 8*	J3-8	J16-8
DIN 7*	J3-10	J16-10
DIN 6*	J3-12	J16-12
DIN 5*	J3-14	J16-14
DIN 4*	J3-16	J16-16
DIN 3*	J3-18	J16-18
DIN 2*	J3-20	J16-20
DIN 1*	J3-22	J16-22
A1	J3-37	J16-37
A2	J3-35	J16-35
A3	J3-33	J16-33
A4	J3-31	J16-31
A5	J3-29	J16-29
A6	J3-27	J16-27
A7	J3-25	J16-25
A8	J3-23	J16-23
OUT *	J3-41	J16-41
IN *	J3-42	J16-42
INTR *	J3-45	J16-45
TTL Ø2	J3-51	J16-51
DI *	J1-17	J16-44
RTS	J1-31	J16-46
DTR	J1-33	J16-48
DO-A	J1-9	J16-40
DO-B	J1-11	J1-9
CTS	J1-49	J16-49
DSR	J1-47	J16-47
RST *	J3-53	J16-53
IREQ *	J3-47	J16-45

APPENDIX B

MAIN PROGRAM AND SUBROUTINE SOFTWARE

Appendix B contains the main program and subroutine software developed for the operation of the automated moisture monitoring system.

HEXADECIMAL			MNEMONIC			TITLE / DATE
PAGE ADR	LINE ADR	INSTR	LABEL	INSTR	MODIFIER	COMMENTS
00	0 0	31		LXI	SP	Initialize stack pointer
	1	CD			CD	
	2	33			33	
	3	01		LXI	BC	Load counter value for USART clock to
	4	28			28	operate at 110 Baud rate
	5	02			02	
	6	CD		JSR	UN(TIMESET)	Subroutine to set up timer mode for
	7	10			10	USART clock
	8	00			00	
	9	CD		JSR	UN(MODESET)	Subroutine to set up USART mode
	A	20			20	
	B	00			00	
	C	C3		JMP	UN(SYSTEM)	Jump to beginning of monitoring system
	D	00			00	software
	E	08			08	
	F	00		NOP		
00	1 0	3E	TIMESET	LDA	I	Load Control Word to set up USART clock, B6 is
	1	B6			B6	select counter 2, Read/Load LS Byte first then MS Byte,
	2	D3		OUT		square wave rate generator, Binary counter 16 bits
	3	27			27	and output to control word register of prog. interval timer
	4	79		LDA	C	Load counter value to USART clock
	5	D3		OUT		Count of 0228
	6	26			26	
	7	78		LDA	B	
	8	D3		OUT		
	9	26			26	
	A	C9		RET	UN	Return to calling routine
	B					
	C					
	D					
	E					
	F					

HEXADECIMAL			MNEMONIC			TITLE / DATE
PAGE ADR	LINE ADR	INSTR	LABEL	INSTR	MODIFIER	COMMENTS
00	2 0	3E	MODESET	LDA	I	Set up USART mode (CE)
	1	CE			CE	
	2	D3		OUT		
	3	21			21	
	4	3E		LDA	I	Send USART command (37)
	5	37			37	
	6	D3		OUT		
	7	21			21	
	8	C9		RET	UN	Return to calling routine
	9					
	A					
	B					
	C					
	D					
	E					
	F					
00	3 0					
	1					
	2					
	3					
	4					
	5					
	6					
	7					
00	3 8	C3		JMP	UN(INTR-RESET)	Jump to interrupt handler
	9	35			85	
	A	15			15	
	B					
	C					
	D					
	E					
	F					

TITLE Main program DATE

HEXADECIMAL PAGE ADR	LINE ADR	INSTR	MNEMONIC LABEL	INSTR	MODIFIER	COMMENTS
08	0 0	21	SYSTEM	LPI	HL	
	1	00			00	Copy initial values from ROM
	2	16			16	into RAM
	3	11		LPI	DE	
	4	00			00	
	5	30			30	
	6	06		LDB	F	
	7	00			00	
	8	CD		JSR	UN(COPY)	
	9	40			40	
	A	00			00	
	B	21		LPI	HL	
	C	00			00	Copy initial values from ROM
	D	17			17	into RAM
	E	11		LPI	DE	
	F	00			00	
08	1 0	31			31	
	1	06		LDB	1	
	2	0F			0F	
	3	CD		JSR	UN(COPY)	
	4	40			40	
	5	00			00	
	6	21		LPI	HL	Load starting address for question
	7	30			30	asking the date
	8	10			10	
	9	CD		JSR	UN(BUFOUT)	Output question to DEC-writer
	A	10			10	
	B	0C			0C	
	C	21		LPI	HL	Load starting memory address where date
	D	32			32	will be stored in ASCII
	E	30			30	
	F	CD		JSR	UN(READ-DEC-ASCII)	Input data from DEC-writer

TITLE DATE

HEXADECIMAL PAGE ADR	LINE ADR	INSTR	MNEMONIC LABEL	INSTR	MODIFIER	COMMENTS
28	2 0	28			28	in ASCII form
	1	0C			0C	
	2	21		LPI	HL	Load starting address for question asking
	3	16			16	for the beginning time
	4	10			10	
	5	CD		JSR	UN(BUFOUT)	Output question to DEC-writer
	6	10			10	
	7	0C			0C	
	8	21		LPI	HL	Load starting memory address where time
	9	0D			0D	will be stored
	A	30			30	
	B	CD		JSR	UN(READ-DEC-ASCII)	Input data from DEC-writer in
	C	28			28	ASCII form
	D	0C			0C	
28	2 E	21	HI-INDIV-READ	LPI	HL	Load starting address for question asking for
	F	30			30	the upper limit on indiv MC reading
28	3 0	10			10	
	1	CD		JSR	UN(BUFOUT)	Output question to DEC-writer
	2	10			10	
	3	0C			0C	
	4	21		LPI	HL	Load starting address for where upper limit
	5	50			50	on indiv MC will be stored in actual form
	6	30			30	
	7	CD		JSR	UN(READ-DEC-ACT)	Input data from DEC-writer in
	8	30			30	actual form
	9	0C			0C	
	A	21		LPI	HL	Load starting address for question asking for
	B	95			95	the lower limit for indiv MC reading
	C	10			10	
	D	CD		JSR	UN(BUFOUT)	Output question to DEC-writer
	E	10			10	
	F	0C			0C	

HEXADECIMAL PAGE ADR	LINE ADR	INSTR	MNEMONIC LABEL	INSTR	MODIFIER	TITLE / DATE COMMENTS
08	4 0	21		LPI	HL	Load starting address where lower limit on
	1	53			53	indiv MC reading will be stored in actual form
	2	30			30	
	3	CD		JSR	UN(READ-DEC-ACT)	Input data from DEC-writer in
	4	30			30	actual form
	5	0C			0C	
	6	C3		JMP	UN(INDIV-CHECK)	Check if lower limit on indiv MC
	7	8A			8A	reading<high limit on indiv MC reading
	8	0E			0E	
08	4 9	21	HI-AVG-READ	LPI	HL	Load starting address for question asking what
	A	CD			CD	is the upper limit for avg MC
	B	10			10	
	C	CD		JSR	UN(BUFOUT)	Output question to DEC-writer
	D	10			10	
	E	0C			0C	
	F	21		LPI	HL	Load starting address where upper limit for
08	5 0	59			59	avg MC will be stored in actual form
	1	30			30	
	2	CD		JSR	UN(READ-DEC-ACT)	Input data from DEC-writer
	3	30			30	in actual form
	4	0C			0C	
	5	21		LPI	HL	Load starting address for question asking what
	6	07			07	is the lower limit for avg MC
	7	11			11	
	8	CD		JSR	UN(BUFOUT)	Output question to DEC-writer
	9	10			10	
	A	0C			0C	
	B	21		LPI	HL	Load starting memory address where lower
	C	5C			5C	limit for avg MC will be stored in actual form
	D	30			30	
	E	CD		JSR	UN(READ-DEC-ACT)	Input data from DEC-writer
	F	30			30	in actual form

HEXADECIMAL PAGE ADR	LINE ADR	INSTR	MNEMONIC LABEL	INSTR	MODIFIER	TITLE / DATE COMMENTS
08	6 0	0C			0C	
	1	C3		JMP	UN(AVG-CHECK)	Check if lower limit for avg
	2	A7			A7	MC<high limit for avg MC
	3	0E			0E	
	4	3E		LDA	I	Load in Run indicator (R) and push onto stack
	5	52			52	
	6	F5		PSP	AF	
08	6 7	21	DIRECTION	LPI	HL	Load starting address for question asking if
	8	41			41	grain is being unloaded
	9	11			11	
	A	CD		JSR	UN(BUFOUT)	Output question to DEC-writer
	B	10			10	
	C	0C			0C	
	D	CD		JSR	UN(CIN)	Read in data from DEC-writer
	E	01			01	
	F	0C			0C	
08	7 0	CD		JSR	UN(COUT)	Echo data back to DEC-writer
	1	10			10	
	2	0C			0C	
	3	E6		AND	I	Set MS bit of data low and
	4	7F			7F	push data onto stack
	5	F5		PSP	AF	
08	7 6	CD	CYCLE	JSR	UN(CIN)	Read in data from DEC-writer
	7	01			01	
	8	0C			0C	
	9	CD		JSR	UN(COUT)	Echo data back to DEC-writer
	A	10			10	
	B	0C			0C	
	C	FE		CPA	I	Check for carriage return, keep bringing
	D	0D			0D	in data until CR occurs.
	E	C2		JMP	ZO (CYCLE)	
	F	76			76	

HEXADECIMAL PAGE ADR	LINE ADR	INSTR	MNEMONIC LABEL	INSTR	MODIFIER	TITLE / DATE COMMENTS
08	8 0	0A			08	
	1	F1		PLP	AF	Load accumulator with direction of grain
	2	F5		PSP	AF	
	3	FE		CPA	I	If grain isn't being unloaded jump to
	4	4E			4E	appropriate questions
	5	CA		JMP	ZI(NO)	
	6	A0			A0	
	7	08			08	
	8	FE		CPA	I	If grain is being unloaded jump to
	9	59			59	appropriate questions
	A	CA		JMP	ZI(YES)	
	B	91			91	
	C	08			08	
	D	F1		PLP	AF	Pull direction byte off stack and since it
	E	C3		JMP	UN(DIRECTION)	is neither Y or N jump back to question
	F	67			67	asking if grain is being unloaded
08	9 0	08			08	
08	9 1	21	YES	LPI	HL	Load starting address for question asking
	2	5E			5E	what bin grain is going into
	3	11			11	
	4	CD		JSR	UN(BUFOUT)	Output question to DEC-Writer
	5	1D			1D	
	6	0C			0C	
	7	21		LPI	HL	Load starting address where bin number
	8	36			36	that grain is going into will be stored
	9	30			30	
	A	CD		JSR	UN(READ-DEC-ASCII)	Bring in data from DEC-Writer and
	B	28			28	store it in ASCII form
	C	0C			0C	
	D	C3		JMP	UN(CONTINUE)	Jump to "CONTINUE" part of main program
	E	2C			2C	
	F	09			09	

HEXADECIMAL PAGE ADR	LINE ADR	INSTR	MNEMONIC LABEL	INSTR	MODIFIER	TITLE / DATE COMMENTS
08	A 0	21	NO	LPI	HL	Load starting address for question asking
	1	80			80	what grain is being loaded onto.
	2	11			11	
	3	CD		JSR	UN(BUFOUT)	Output question to DEC-writer
	4	1D			1D	
	5	0C			0C	
	6	21		LPI	HL	Load starting address where information telling
	7	36			36	what grain is being loaded onto will be stored
	8	30			30	
	9	CD		JSR	UN(READ-DEC-ASCII)	Bring in data from DEC-writer and
	A	28			28	store it in ASCII form
	B	0C			0C	
	C	21		LPI	HL	Load starting address for question asking
	D	A5			A5	how long it will take to load grain
	E	11			11	
	F	CD		JSR	UN(BUFOUT)	Output question to DEC-Writer
08	B 0	1D			1D	
	1	0C			0C	
	2	21		LPI	HL	Load starting address where how long to
	3	A4			A4	load grain will be stored in ASCII form
	4	30			30	
	5	CD		JSR	UN(READ-DEC-ASCII)	Read data in from DEC-Writer and store
	6	28			28	it in ASCII form
	7	0C			0C	
	8	21		LPI	HL	Load starting address where no. of predicted
	9	A4			A4	readings is stored in ASCII
	A	30			30	
	B	11		LPI	DE	Load starting address where no. of
	C	AD			AD	predicted readings will be stored in Calculator form
	D	30			30	
	E	06		LDB	I	Load counter for number of bytes to convert
	F	05			05	

ORIGINAL PAGE IS
OF POOR QUALITY

HEXADECIMAL PAGE ADR	LINE ADR	INSTR	LABEL	MNEMONIC INSTR	MODIFIER	TITLE / DATE COMMENTS
08	C 0	CD		JSR	UP(ASCII-CAL)	Convert ASCII numbers into calculator
	1	57			57	form
	2	00			00	
	3	21		LPI	HL	Load starting address where no. of predicted
	4	A4			A4	readings is stored in ASCII
	5	30			30	
	6	11		LPI	DE	Load starting address where no. of predicted
	7	C2			C2	readings will be stored in calculator form
	8	30			30	
	9	06		LDB	I	Load counter for number of bytes to convert
	A	05			05	
	B	CD		JSR	UN(ASCII-CAL)	Conver data from ASCII form into
	C	57			57	calculator form
	D	00			00	
08	C E	21	DESIR-FIN-AVG	LPI	HL	Load starting address of question asking
	F	E9			E9	what is the desired final avg MC
08	D 0	11			11	
	1	CD		JSR	UN(BUFOUT)	Output question to DEC-writer
	2	1D			1D	
	3	0C			0C	
	4	21		LPI	HL	Load starting address where desired final
	5	A9			A9	avg. MC will be stored in ASCII form
	6	30			30	
	7	CD		JSR	UN(READ-DEC-ASCII)	Input data from the DEC-writer
	8	2B			2B	in ASCII form
	9	0C			0C	
	A	21		LPI	HL	Load starting address where desi final avg
	B	A9			A9	MC is stored in ASCII form
	C	30			30	
	D	11		LPI	DE	Load starting address where desired final avg
	E	E2			82	MC will be stored in Calculator form
	F	30			30	

HEXADECIMAL PAGE ADR	LINE ADR	INSTR	LABEL	MNEMONIC INSTR	MODIFIER	TITLE / DATE COMMENTS
08	E 0	06		LDB	I	Load counter for number of bytes to be
	1	04			04	converted
	2	CD		JSR	UN(ASCII-CAL)	Convert data from ASCII form into
	3	57			57	Calculator form
	4	00			00	
	5	C3		JMP	UN(DESIR-AVG-TEST)	Check if desired final avg MC is within
	6	C4			C4	limits for avg MC
	7	0E			0E	
08	E 8	21	READ-INTERVAL	LPI	HL	Load starting address for question asking how
	9	1D			1D	many reading intervals between calculations
	A	13			13	for remaining MC
	B	CD		JSR	UN(BUFOUT)	Ouptut question to DEC-writer
	C	1D			1D	
	D	0C			0C	
	E	21		LPI	HL	Load address where no. of readings interval
	F	F7			F7	will be stored in decimal form
08	F 0	30			30	
	1	CD		JSR	UN(READ-DEC-COUNT)	Input data in decimal form
	2	6C			6C	
	3	0C			0C	
	4	21		LPI	HL	Load starting address where readings interval
	5	F7			F7	is stored in decimal form
	6	30			30	
	7	11		LPI	DE	Load starting address where readings interval
	8	F8			F8	will be stored in hexadecimal form
	9	30			30	
	A	CD		JSR	UN(DECIMAL-HEX)	Convert decimal byte into hexadecimal byte
	B	6F			6F	
	C	0E			0E	
	D	21		LPI	HL	Have no. of readings interval stored in two
	E	F8			F8	memory spaces.
	F	30			30	

HEXADECIMAL			MNEMONIC			TITLE / DATE
PAGE ADR	LINE ADR	INSTR	LABEL	INSTR	MODIFIER	COMMENTS
09	0 0	7E		LDAM	HL	
	1	23		INCP	HL	
	2	77		STAM	HL	
09	0 3	CD	INIT-CALC	JSR	UN(RESET)	REset calculator card
	4	9E			9E	
	5	0C			0C	
	6	01		LPI	BC	Load starting address where calculator
	7	80			80	instructions are stored
	8	15			15	
	9	16		LDD	I	Set up counter for number of bytes to be
	A	03			03	sent to calculator card
	B	CD		JSR	UN(LOAD-W-COUNT)	Send desired number of data bytes
	C	AF			AF	to calculator card
	D	0C			0C	
	E	01		LPI	BC	Load starting address where data to be
	F	AD			AD	sent to calculator card is stored.
09	1 0	30			30	Calculate desired total of MC readings
	1	16		LDD	I	Set up counter for number of bytes to
	2	08			08	be sent to calculator card
	3	CD		JSR	UN(LOAD-W-COUNT)	Send desired number of data
	4	AF			AF	bytes to calculator card
	5	0C			0C	
	6	CD		JSR	UN(OUT)	Instruct calculator card that
	7	CB			CB	data is to be read from it
	8	0C			0C	
	9	11		LPI	DE	Load starting address where desired total of
	A	B8			B8	MC readings will be stored in answer form
	B	30			30	
	C	06		LDB	I	Set up counter for number of bytes to be
	D	0A			0A	read from calculator card
	E	CD		JSR	UN(READ-CAL)	Read desired total of MC readings from
	F	EA			EA	calculator card

HEXADECIMAL			MNEMONIC			TITLE / DATE
PAGE ADR	LINE ADR	INSTR	LABEL	INSTR	MODIFIER	COMMENTS
09	2 0	EC			0C	
	1	21		LPI	HL	Load starting address of desired total of
	2	B8			B8	MC readings in answer form
	3	30			30	
	4	11		LPI	DE	Load starting address where desired total of
	5	D4			D4	MC readings are to be stored in calculator form
	6	30			30	
	7	06		LDB	I	Set up counter for number of bytes of data
	8	0A			0A	to be converted
	9	CD		JSR	UN(ANS-CAL)	Convert data from answer form into
	A	72			72	Calculator form
	B	00			0D	
09	2 C	21	CONTINUE	LPI	HL	Load starting address of title heading
	D	D2			D2	
	E	12			12	
	F	CD		JSR	UN(BUFOUT)	Output title heading to CEC-writer
09	3 0	1D			1D	
	1	0C			0C	
09	3 2	3E	1-MIN-INTR	LDA	I	Output control word (30) to control word register
	3	30			30	It selects counter 0, read/load LSB first
	4	D3		OUT		then MSB, interrupt on terminal count
	5	27			27	16 bit Binary counter
	6	3E		LDA	I	
	7	70			70	Send to counter 0, LSB first then MSB
	8	D3		OUT		
	9	24			24	Count of 1770
	A	3E		LDA	I	
	B	17			17	
	C	D3		OUT		
	D	24			24	
	E	3E		LDA	I	Output control word (76) to control word register
	F	76			76	It selects counter 1, Read/Load. LSB first, then MSB

HEXADECIMAL PAGE ADR	HEXADECIMAL LINE ADR	HEXADECIMAL INSTR	MNEMONIC LABEL	MNEMONIC INSTR	MNEMONIC MODIFIER	TITLE / DATE / COMMENTS
09	4 0	D3		OUT		square wave rate generator acting as clock
	1	27			27	for counter 0, 16 bit binary counter
	2	3E		LDA	1	
	3	10			10	Send to counter 1, LSB first then MSB
	4	D3		OUT		
	5	25			25	
	6	3E		LDA	1	Count of 2710
	7	27			27	
	8	D3		OUT		
	9	25			25	
	A	FB		EIN		Enable interrupt
09	4 B	3E	LOAD-GRAIN	LDA	1	
	C	FE			FE	Turn on #1 triac to load grain sample
	D	D3		OUT		into test cell of moisture meter
	E	01			01	
09	4 F	01	1-SEC-DELAY	LPI	BC	Load register B, C, and D with count for
09	5 0	04			04	1 sec delay
	1	02			02	
	2	16		LDD	1	
	3	67			67	
09	5 4	15	DELAY 1	DEC	D	Decrement registers to 0 to allow
	5	C2		JMP	ZO (DELAY 1)	#1 Triac to remain on for 1 sec
	6	54			54	
	7	09			09	
	8	0D		DEC	C	
	9	C2		JMP	ZO(DELAY 1)	
	A	54			54	
	B	09			09	
	C	05		DEC	B	
	D	C2		JMP	ZO(DELAY 1)	
	E	54			54	
	F	09			09	

HEXADECIMAL PAGE ADR	HEXADECIMAL LINE ADR	HEXADECIMAL INSTR	MNEMONIC LABEL	MNEMONIC INSTR	MNEMONIC MODIFIER	TITLE / DATE / COMMENTS
09	5 0	3E		LDA	1	
	1	FF			FF	Turn #1 Triac off
	2	D3		OUT		
	3	01			01	
0 9	6 4	01	15-SEC-DELAY	LPI	BC	Load registers B,C and D with count
	5	33			33	for 15 sec delay
	6	10			10	
	7	16		LDD	1	
	8	FF			FF	
09	6 9	15	DELAY 2	DEC	D	Decrement registers to 0 to allow
	A	C2		JMP	ZO(DELAY 2)	15 sec delay before reading is taken
	B	69			69	from moisture meter
	C	09			09	
	D	0D		DEC	C	
	E	C2		JMP	ZO(DELAY 2)	
	F	69			69	
09	7 0	09			09	
	1	05		DEC	B	
	2	C2		JMP	ZO (DELAY 2)	
	3	69			69	
	4	09			09	
09	7 5	21	SAMPLE	LPI	HL	Load starting address where indiv MC
	6	56			56	reading in actual form will be stored
	7	30			30	
	8	CD		JSR	UN(READ-MC-ACT)	Read data from moisture meter and store
	9	54			54	in actual form
	A	0C			0C	
	B	C3		JMP	UN(HI-VALID-TEST)	Check if reading is too high or too
	C	04			04	low to be valid
	D	0F			0F	
09	7 E	21	SAMPLE-CONT	LPI	HL	Load starting address where indiv
	F	56			56	MC is stored in actual form

HEXADECIMAL			MNEMONIC			TITLE DATE
PAGE ADR	LINE ADR	INSTR	LABEL	INSTR	MODIFIER	COMMENTS
09	8 0	30			30	
	1	11		LPI	DE	Load starting address where indiv
	2	1C			1C	MC is stored in ASCII form
	3	30			30	
	4	CD		JSR	UN(ACT-ASCII)	Convert data from actual form into
	5	C3			C3	ASCII form
	6	0D			0D	
	7	21		LPI	HL	Load starting address where indiv
	8	56			56	MC is stored in actual form
	9	30			30	
	A	11		LPI	DE	Load starting address where indiv
	B	66			66	MC will be stored in calculator form
	C	30			30	
	D	CD		JSR	UN(ACT-CAL)	Convert data from actual form
	E	DC			DC	into calculator form
	F	0D			0D	
09	9 0	CD		JSR	UN(RESET)	REset calculator card
	1	9E			9E	
	2	0C			0C	
	3	01		LPI	BC	Load starting address where calculator
	4	80			80	instructions are stored
	5	15			15	
	6	16		LDD	I	Load counter for number of bytes of instructions
	7	03			03	to send to calculator card
	8	CD		JSR	UN(LOAD W-Count)	Send instructions to calculator card
	9	AF			AF	
	A	0C			0C	
	B	01		LPI	BC	Load starting address where data to be
	C	66			66	sent to calculator card is stored
	D	30			30	calculate total of MC readings
	E	16		LDD	I	Load counter for number of bytes of data
	F	0F			0F	to be sent to calculator card

HEXADECIMAL			MNEMONIC			TITLE DATE
PAGE ADR	LINE ADR	INSTR	LABEL	INSTR	MODIFIER	COMMENTS
09	A 0	CD		JSR	UN(LOAD-W-COUNT)	Send desired number of data bytes to
	1	AF			AF	calculator card
	2	0C			0C	
	3	CD		JSR	UN(OUT)	Instruct calculator card that the
	4	C8			C8	data it has is to be read
	5	0C			0C	
	6	11		LPI	DE	Load starting address where total of MC
	7	75			75	readings will be stored in answer form
	8	30			30	
	9	06		LDB	I	Load counter for number of bytes to be read
	A	0A			0A	from calculator card
	B	CD		JSR	UN(READ-CAL)	Read total of MC readings from calculator
	C	EA			EA	card
	D	0C			0C	
	E	21		LPI	HL	Load starting address where total of MC
	F	75			75	readings is stored in answer form
09	B 0	30			30	
	1	11		LPI	DE	Load starting address where total of MC
	2	6B			6B	readings will be stored in calculator form
	3	30			30	
	4	06		LDB	I	Load counter for number of bytes to be
	5	0A			0A	converted
	6	CD		JSR	UN(ANS-CAL)	Convert data from answer form into
	7	72			72	calculator form
	8	0D			0D	
	9	21		LPI	HL	Load starting address where total of
	A	75			75	MC readings will be stored in answer form
	B	30			30	
	C	11		LPI	DE	Load starting address where total of MC
	D	8F			8F	readings will be stored in calculator form
	E	30			30	
	F	26		LDE	I	Load counter for number of bytes of data

HEXADECIMAL			MNEMONIC			TITLE / DATE
PAGE ADR	LINE ADR	INSTR	LABEL	INSTR	MODIFIER	COMMENTS
09	C 0	0A			0A	to be converted
	1	CD		JSR	UN(ANS-CAL)	Convert data from answer form into
	2	72			72	Calculator form
	3	00			00	
	4	21		LPI	HL	Load starting address where total of MC
	5	75			75	readings is stored in answer form
	6	30			30	
	7	11		LPI	DE	Load starting address where total of MC
	8	DE			DE	readings is to be stored in calculator form
	9	30			30	
	A	06		LDB	I	Load counter for number of bytes of data to
	B	0A			0A	be converted
	C	CD		JSR	UN(ANS-CAL)	Convert data from answer form into
	D	72			72	Calculator form
	E	00			00	
	F	CD		JSR	UN(RESET)	Reset calculator card in preparation
09	D 0	9E			9E	for another calculation
	1	0C			0C	
09	D 2	01		LPI	BC	Load starting address of instructions
	3	80			80	to be sent to calculator card
	4	15			15	
	5	16		LDD	I	Load counter with number of bytes of
	6	03			03	instructions to send to calculator card
	7	CD		JSR	UN(LOAD-W-COUNT)	Send instructions to calculator card
	8	AF			AF	
	9	0C			0C	
	A	01		LPI	BC	Load starting address where data to calculate
	B	7F			7F	number of readings taken is stored
	C	30			30	
	D	16		LDD	I	Load counter for number of bytes to be
	E	0A			0A	sent to calculator card
	F	CD		JSR	UN(LOAD-W-COUNT)	Send desired number of data bytes to calculator

HEXADECIMAL			MNEMONIC			TITLE / DATE
PAGE ADR	LINE ADR	INSTR	LABEL	INSTR	MODIFIER	COMMENTS
09	E 0	AF			AF	card
	1	0C			0C	
	2	CD		JSR	UN(OUT)	Instruct calculator card that its data
	3	CB			CB	is to be read
	4	0C			0C	
	5	11		LPI	DE	Load starting address where number of
	6	89			89	readings taken is to be stored in ahswer form
	7	30			30	
	8	06		LDB	I	Load counter for number of bytes to be
	9	06			06	read from calculator card
	A	CD		JSR	UN(READ-CAL)	Read number of readings taken from
	B	EA			EA	calculator card
	C	0C			0C	
	D	21		LPI	HL	Load starting address where number of
	E	89			89	readings taken is stored in answer form
	F	30			30	
09	F 0	11		LPI	DE	Load starting address where number of
	1	7F			7F	readings taken is to be stored in calculator form
	2	30			30	
	3	06		LDB	I	Load counter for number of data bytes
	4	06			06	to be converted
	5	CD		JSR	UN(ANS-CAL)	Convert data from answer form into
	3	72			72	calculator form
	7	00			00	
	8	21		LPI	HL	Load starting address where number of
	9	89			89	readings taken is stored in answerform
	A	30			30	
	B	11		LPI	DE	Load starting address where number of
	C	99			99	readings taken is to be stored in calculator form
	D	30			30	
	E	06		LDB	I	Load counter for number of data bytes
	F	06			06	to be converted

HEXADECIMAL			MNEMONIC			TITLE DATE
PAGE ADR	LINE ADR	INSTR	LABEL	INSTR	MODIFIER	COMMENTS
0A	0 0	CD		JSR	UN(ANS-CAL)	Convert data from answer form into
	1	72			72	calculator form
	2	00			00	
	3	21		LPI	HL	Load starting address where number of readings
	4	89			89	taken is stored in answer form
	5	30			30	
	6	11		LPI	DE	Load starting address when number of readings
	7	C8			C8	taken is to be stored in calculator form
	8	30			30	
	9	06		LDB	1	Load counter for number of data bytes to be
	A	06			06	converted
	B	CD		JSR	UN(ANS-CAL)	Convert data from answer form into
	C	72			72	calculator form
	D	00			00	
	E	CD		JSR	UN(RESET)	Reset calculator in preparation for
	F	9E			9E	another calculation
0A	1 0	0C			0C	
	1	01		LPI	BC	Load beginning address where calculator
	2	80			80	instructions are stored
	3	15			15	
	4	16		LDD	1	Set up counter for number of instruction
	5	03			03	bytes to be sent to calculator
	6	CD		JSR	UN(LOAD-W-COUNT)	Send desired number of data bytes to
	7	AF			AF	calculator
	8	0C			0C	
	9	01		LPI	BC	Load starting address for data to be used
	A	8F			8F	in calculating avg MC
	B	30			30	
	C	16		LDD	1	Load counter for number of data bytes to
	D	10			10	be sent to calculator
	E	CD		JSR	UN(LOAD-W-COUNT)	Send desired number of data bytes to
	F	AF			AF	calculator

HEXADECIMAL			MNEMONIC			TITLE DATE
PAGE ADR	LINE ADR	INSTR	LABEL	INSTR	MODIFIER	COMMENTS
0A	2 0	0C			0C	
	1	CD		JSR	UN(OUT)	Instruct calculator that data it has is
	2	C8			C8	to be read
	3	0C			0C	
	4	11		LPI	DE	Load starting address where avg MC is to
	5	9F			9F	be stored in answer form
	6	30			30	
	7	06		LDB	1	Load counter for number of data bytes
	8	05			05	to be read
	9	CD		JSR	UN(READ-CAL)	Read avg MC from calculator
	A	EA			EA	
	B	0C			0C	
	C	21		LPI	HL	Load starting address where avg MC is
	D	9F			9F	stored in answer form
	E	30			30	
	F	11		LPI	DE	Load starting address where avg MC is
0A	3 0	62			62	to be stored in calculator form
	1	30			30	
	2	06		LDB	1	Load counter for number of data bytes
	3	05			05	to be converted
	4	CD		JSR	UN(AVG-ANS-CAL)	Convert avg MC from answer form
	5	22			22	into calculator form
	6	0E			0E	
	7	21		LPI	HL	Load starting address where avg MC value
	8	62			62	is stored in calculator form
	9	30			30	
	A	11		LPI	DE	Load starting address where avg MC value
	B	28			28	is to be stored in ASCII form
	C	30			30	
	D	06		LDB	1	Load counter for number of data bytes
	E	04			04	to be converted
	F	CD		JSR	UN(CAL-ASCII)	Convert data from calculator form into ASCII form

HEXADECIMAL			MNEMONIC			TITLE DATE
PAGE ADR	LINE ADR	INSTR	LABEL	INSTR	MODIFIER	COMMENTS
0A	4 0	F0			F0	
	1	00			00	
	2	21		LPI	HL	Lo d starting address where data record
	3	00			00	is stored
	4	30			30	
	5	CD		JSR	UN(BUFOUT)	Output data record to DEC-writer
	6	10			10	
	7	0C			0C	
	8	21		LPI	HL	Load starting address where avg MC
	9	62			62	value is stored in calculator form
	A	30			30	
	B	11		LPI	DE	Load starting address where avg MC
	C	5F			5F	value is to be stored in actual form
	D	30			30	
	E	06		LDB	I	Load counter for number of data bytes to be
	F	04			04	converted
0A	5 0	CD		JSR	UN(CAL-ACT)	Convert data from calculator form into
	1	0B			0B	actual form
	2	0E			0E	
	3	21		LPI	HL	Load starting address where upper limit on
	4	50			50	indiv MC reading is stored in actual form
	5	30			30	
	6	11		LPI	DE	Load starting address where indiv MC
	7	56			56	reading is stored in actual form
	8	30			30	
	9	06		LDB	I	Load counter for number of data bytes to be
	A	03			03	compared
	B	CD		JSR	UN(HI-INDIV-TEST)	Check indiv MC reading for exceeding
	C	3C			3C	high limit
	D	0F			0F	
	E	21		LPI	HL	Load starting address where lower limit on
	F	53			53	indiv MC reading is stored in actual form

HEXADECIMAL			MNEMONIC			TITLE DATE
PAGE ADR	LINE ADR	INSTR	LABEL	INSTR	MODIFIER	COMMENTS
0A	6 0	30			30	
	1	11		LPI	DE	Load starting address where indiv MC reading
	2	56			56	is stored in actual form
	3	30			30	
	4	06		LDB	I	Load counter for number of data bytes to
	5	03			03	be compared
	6	CD		JSR	UN(LO-INDIV-TEST)	Check indiv MC reading for being below
	7	40			40	low limit
	8	0F			0F	
	9	21		LPI	HL	Load starting address where upper limit
	A	59			59	on avg MC value is stored in actual form
	B	30			30	
	C	11		LPI	DE	Load starting address where avg MC
	D	5F			5F	value is stored in actual form
	E	30			30	
	F	06		LDB	I	Load counter for number of data bytes
0A	7 0	03			03	to be compared
	1	CD		JSR	UN(HI AVG-TEST)	Check avg MC value for exceeding
	2	5E			5E	high limit
	3	0F			0F	
	4	21		LPI	HL	Load starting address where lower limit on
	5	5C			5C	avg MC value is stored in actual form
	6	30			30	
	7	11		LPI	DE	Load starting address where avg MC
	8	5F			5F	value is stored in actual form
	9	30			30	
	A	06		LDB	I	Load counter for number of data bytes
	B	03			03	to be compared
	C	CD		JSR	UN(LO-AVG-TEST)	Check avg MC value for being lower
	D	6F			6F	than low limit
	E	0F			0F	
	F	F1		PLP	AF	Put destination indicator into accumulator

HEXADECIMAL			MNEMONIC			TITLE DATE
PAGE ADR	LINE ADR	INSTR	LABEL	INSTR	MODIFIER	COMMENTS
0A	8 0	F5		PSP	AF	
	1	FE		CPA	I	If grain is being unloaded (Y indicator)
	2	59			59	don't calculate desired final avg MC so
	3	CA		JMP	21(UNLOAD-GRAIN)	jump to UNLOAD-GRAIN routine
	4	01			01	
	5	08			08	
	6	21		LPI	HL	Load address where counter for no. of
	7	F9			F9	readings until calculation of required remaining
	8	30			30	MC is stored
	9	7E		LDAN	HL	Load counter value
	A	3D		DEC	A	Decrement counter
	B	77		STAN	HL	and store new value back into memory
	C	C2		JMP	20(UNLOAD-GRAIN)	If it isn't time to do calculation go
	D	01			01	to UNLOAD-GRAIN routine
	E	08			08	
	F	CD		JSR	UN(RESET)	Reset calculator in preparation for doing
0A	9 0	9E			9E	another calculation
	1	0C			0C	
	2	01		LPI	BC	Load starting address where calculator
	3	80			80	instructions are stored
	4	15			15	
	5	16		LDD	I	Load counter for number of data bytes to
	6	03			03	be sent to calculator
	7	CD		JSR	UN(LOAD-W-COUNT)	Send desired number of data bytes to calculator
	8	AF			AF	
	9	0C			0C	
	A	01		LPI	BC	Load starting address where data used to
	B	C2			C2	calculate no. of remaining readings is
	C	30			30	stored
	D	16		LDD	I	Load counter for number of data bytes to
	E	0C			0C	be sent to calculator
	F	CD		JSR	UN(LOAD-W-COUNT)	Send desired number of data bytes to calculator

HEXADECIMAL			MNEMONIC			TITLE DATE
PAGE ADR	LINE ADR	INSTR	LABEL	INSTR	MODIFIER	COMMENTS
0A	A 0	AF			AF	
	1	0C			0C	
	2	CD		JSR	UN(OUT)	Instruct calculator that its data is to
	3	C8			C8	be read
	4	0C			0C	
	5	11		LPI	DE	Load starting address where no. of readings
	6	CE			CE	remaining will be stored in answer form
	7	30			30	
	8	06		LDB	I	Load counter for number of data bytes to
	9	06			06	be read from calculator
	A	CD		JSR	UN(READ-CAL)	Read desired number of data bytes from calculator
	B	EA			EA	
	C	0C			0C	
	D	21		LPI	HL	Load starting address where no. of readings
	E	CE			CE	remaining is stored in answer form
	F	30			30	
0A	B 0	11		LPI	DE	Load starting address where no. of readings
	1	E8			E8	remaining is to be stored in calculator form
	2	30			30	
	3	06		LDB	I	Load counter for number of data bytes to
	4	06			06	be converted
	5	CD		JSR	UN(ANS-CAL)	Convert data from answer form into
	6	72			72	calculator form
	7	00			00	
	8	C3		JMP	UN(CHECK-READING)	Jump to routine to check if number
	9	80			80	of readings remaining is zero
	A	0F			0F	
0A	B B	CD	DESIRED-MC	JSR	UN(RESET)	Reset calculator in preparation for another
	C	9E			9E	calculation
	D	0C			0C	
	E	01		LPI	BC	Load starting address where calculator
	F	80			80	instructions are stored

HEXADECIMAL			MNEMONIC			TITLE DATE
PAGE ADR	LINE ADR	INSTR	LABEL	INSTR	MODIFIER	COMMENTS
0A	C 0	15			15	
	1	16		LDD	I	Load counter for number of instruction
	2	03			03	bytes to be sent to calculator
	3	CD		JSR	UN(LOAD-W-COUNT)	Send desired number of instructions
	4	AF			AF	to calculator
	5	OC			OC	
	6	01		LPI	BC	Load starting address where data to be
	7	D4			D4	used in calculating desired MC for
	8	30			30	remaining grain is stored
	9	16		LDD	I	Load counter for number of data bytes
	A	1A			1A	to be sent to calculator
	B	CD		JSR	UN(LOAD-W-COUNT)	Send desired number of data bytes
	C	AF			AF	to calculator
	D	OC			OC	
	E	CD		JSR	UN(OUT)	Instruct calculator that its
	F	CB			CB	data is to be read
0A	D 0	OC			OC	
	1	11		LPI	DE	Load starting address where desired MC for
	2	EE			EE	remaining grain is to be stored in answer form
	3	30			30	
	4	06		LDB	I	Load counter for number of data bytes
	5	05			05	to be read
	6	CD		JSR	UN(READ-CAL)	Read desired MC for remaining grain
	7	EA			EA	from calculator
	8	OC			OC	
	9	21		LPI	HL	Load starting address where desired MC for
	A	EE			EE	remaining grain is stored in answer form
	B	30			30	
	C	11		LPI	DE	Load starting address where desired MC for
	D	F3			F3	remaining grain is to be stored in calculator form
	E	30			30	
	F	06		LDB	I	Load counter for number of data bytes to

HEXADECIMAL			MNEMONIC			TITLE DATE
PAGE ADR	LINE ADR	INSTR	LABEL	INSTR	MODIFIER	COMMENTS
0A	E 0	05			05	be converted
	1	CD		JSR	UN(AVG-ANS-CAL)	Convert data from answer form into
	2	22			22	Calculator form
	3	OE			OE	
	4	21		LPI	HL	Load starting address where desired remaining
	5	F3			F3	avg MC is stored in calculator form
	6	30			30	
	7	11		LPI	DE	Load starting address where desired remaining
	8	08			08	avg MC is to be stored in ASCII form
	9	31			31	
	A	06		LDB	I	Load counter for number of data bytes to
	B	04			04	be converted
	C	CD		JSR	UN(CAL-ASCII)	Convert data from calculator form into
	D	F0			F0	ASCII form
	E	0D			0D	
	F	21		LPI	HL	Load starting address for message telling
0A	F 0	C2			C2	the desired remaining MC
	1	13			13	
	2	CD		JSR	UN(BUFOUT)	Output message to DEC-Writer
	3	10			10	
	4	OC			OC	
	5	21		LPI	HL	Load starting address for desired remaining
	6	08			08	MC value in ASCII
	7	31			31	
	8	CD		JSR	UN(BUFOUT)	Output value to DEC-Writer
	9	10			10	
	A	OC			OC	
	B	21		LPI	HL	Load address where counter for number of
	C	FB			FB	readings between each calculation of remaining
	D	30			30	avg MC to obtain desired final avg MC is stored
	E	7E		LDAN	HL	Reload counter that was used to determine
	F	23		INCP	HL	when to make calculation

Hexadecimal Page Adr	Line Adr	Instr	Mnemonic Label	Instr	Modifier	Title / Comments
08	0 0	77		STAN	HL	
08	0 1	3E	UNLOAD-GRAIN	LDA	I	Turn #2 triac on to open bottom of
	2	FD			FD	test cell on moisture meter to allow
	3	D3		OUT		grain to fall out of test cell
	4	01			01	
08	0 5	01	1-SEC-DELAY 1	LPI	BC	Load Registers B, C, and D with count to
	6	04			04	cause 1 sec delay
	7	02			02	
	8	16		LDD	I	
	9	67			67	
08	0 A	15	DELAY 3	DEC	D	Decrement registers to zero to
	B	C2		JMP	ZO (DELAY 3)	allow #2 triac to remain on for
	C	0A			0A	1 sec.
	D	08			08	
	E	0D		DEC	C	
	F	C2		JMP	ZO (DELAY 3)	
08	1 0	0A			0A	
	1	08			08	
	2	05		DEC	B	
	3	C2		JMP	ZO(DELAY 3)	
	4	0A			0A	
	5	08			08	
	6	3E		LDA	I	Turn #2 triac off to allow bottom of
	7	FF			FF	test cell to close
	8	D3		OUT		
	9	01			01	
08	1 A	2A	TIMER	LHLD	HL	Put the hours of the time
	B	0D			0D	into Registers B and C
	C	30			30	
	D	45		LDB	L	
	E	4C		LDC	H	
	F	2A		LHLD	HL	Put the minutes value of the time

Hexadecimal Page Adr	Line Adr	Instr	Mnemonic Label	Instr	Modifier	Title / Comments
08	2 0	10			10	into Registers D and E
	1	30			30	
	2	55		LDD	L	
	3	5C		LDE	H	
	4	2A		LHLD	HL	Register H continues ASCII M and Register
	5	13			13	L containes ASCII A or P for AM or PM
	6	30			30	
	7	7B		LDA	E	Change minutes value from two ASCII
	8	E6		AND	I	bytes to one byte with a two digit decimal
	9	0F			0F	value. Store this value in Register E.
	A	5F		LDE	A	
	B	7A		LDA	D	
	C	E6		AND	I	
	D	0F			0F	
	E	07		RLC		
	F	07		RLC		
08	3 0	07		RLC		
	1	07		RLC		
	2	B3		ORA	E	
	3	5F		LDE	A	
	4	79		LDA	C	Change hours value from two ASCII bytes
	5	E6		AND	I	to one byte with a two digit decimal
	6	0F			0F	value. Store this value in Register C.
	7	4F		LDC	A	
	8	78		LDA	B	
	9	E6		AND	I	
	A	0F			0F	
	B	07		RLC		
	C	07		RLC		
	D	07		RLC		
	E	07		RLC		
	F	B1		ORA	C	

HEXADECIMAL PAGE ADR	LINE ADR	INSTR	MNEMONIC LABEL	INSTR	MODIFIER	TITLE / DATE / COMMENTS
08	4 0	4F		LDC	A	
	1	78		LDA	E	Add time interval between readings
	2	C6		ADA	I	(1 min) to previous time
	3	01			01	
	4	27		DAA		
	5	5F		LDE	A	
	6	FE		CPA	I	Check if minutes time ≥ 60
	7	60			60	
	8	DA		JMP	C1(STOR-TIME)	If it isn't jump to routine to store
	9	9E			9E	the time back into memory
	A	08			08	
	B	D6		SUA	I	If it is subtract 60 from minutes value
	C	60			60	to get correct minutes time and store
	D	5F		LDE	A	back into Register E
	E	79		LDA	C	
	F	C6		ADA	I	Add one to the number of hours of the
08	5 0	01			01	time
	1	27		DAA		
	2	4F		LDC	A	
08	5 3	FE	CHECK-13	CPA	I	Check if hours value is 13
	4	13			13	
	5	C2		JMP	ZO(CHECK-12)	If it isn't jump to routine to check
	6	5D			5D	for hours value being 12
	7	08			08	
	8	0E		LDC	I	If hours value is 13 change hours value
	9	01			01	to 01
	A	C3		JMP	UN(STOR-TIME)	and jump to routine to store time
	B	9E			9E	back into memory
	C	08			08	
08	5 D	FE	CHECK-12	CPA	I	Check if hours value is 12
	E	12			12	
	F	C2		JMP	ZO(STOR-TIME)	If it isn't jump to routine to store

HEXADECIMAL PAGE ADR	LINE ADR	INSTR	MNEMONIC LABEL	INSTR	MODIFIER	TITLE / DATE / COMMENTS
08	6 0	9E			9E	time back into memory
	1	08			08	
	2	7D		LDA	L	Load in ASCII A or P
	3	E6		AND	I	Set MS bit to zero since it isn't used
	4	7F			7F	in ASCII coding
	5	FE		CPA	I	Check if time designation is A
	6	41			41	
	7	C2		JMP	ZO(PM)	If time designation isn't A jump to
	8	6F			6F	routine that handles PM time designation
	9	08			08	
	A	2E		LDL	I	If first letter was A change to P
	B	50			50	
	C	C3		JMP	UN(STOR-TIME)	and jump to routine to store time back
	D	9E			9E	into memory
	E	08			08	
08	6 F	2E	PM	LDL	I	Change P to an A for AM
08	7 0	41			41	
	1	44		LDB	H	Load register B from register H
	2	55		LDD	L	Load register D from register L
	3	21		LPI	HL	Bring in second digit of the day of the
	4	06			06	month in ASCII
	5	30			30	
	6	7E		LDA	M	
	7	E6		AND	I	Set meaningless MS bit low
	8	7F			7F	
	9	C6		ADA	I	Add one day to the original value
	A	01			01	
	B	27		DAA		
	C	6F		LDL	A	and store ba[illegible] register L
	D	FE		CPA	I	Check [illegible] indicating first
	E	40			40	day of [illegible] incremented
	F	C2		JMP	ZO(STOR-DATE)	If no A[illegible] ine to store

HEXADECIMAL			MNEMONIC			TITLE DATE
PAGE ADR	LINE ADR	INSTR	LABEL	INSTR	MODIFIER	COMMENTS
08	8 0	97			97	date back into memory
	1	08			08	
	2	3E		LDA	I	Put a ASCII 30 in memory space
	3	30			30	for second digit of the day of the month
	4	21		LPI	HL	
	5	06			06	
	6	30			30	
	7	77		LDM	A	
	8	21		LPI	HL	Load first digit of the day of the month
	9	05			05	
	A	30			30	
	B	7E		LDA	M	
	C	E6		AND	I	Set meaningless MS bit low
	D	7F			7F	
	E	C6		ADA	I	Add one to the original value and store back
	F	01			01	into memory
08	9 0	27		DAA		
	1	77		LDM	A	
	2	60		LDH	B	Put PM or AM back into HL register pair
	3	6A		LDL	D	
	4	C3		JMP	UN(STOR-TIME)	Jump to routine to store the time
	5	9E			9E	back into memory
	6	08			08	
08	9 7	7D	STOR-DATE	LDA	L	Put the second digit of the day of the month
	8	21		LPI	HL	back into memory
	9	06			06	
	A	30			30	
	B	77		LDM	A	
	C	60		LDH	B	Put PM or AM back into HL register pair
	D	6A		LDL	D	
08	9 E	22	STOR-TIME	SHLD	HL	Store PM or AM back into memory space
	F	13			13	

HEXADECIMAL			MNEMONIC			TITLE DATE
PAGE ADR	LINE ADR	INSTR	LABEL	INSTR	MODIFIER	COMMENTS
08	A 0	30			30	
	1	7B		LDA	E	Change first digit of time in minutes
	2	0F		RRC		into ASCII coding
	3	0F		RRC		
	4	0F		RRC		
	5	0F		RRC		
	6	E6		AND	I	
	7	0F			0F	
	8	F6		ORA	I	Store ASCII value of first digit of minutes
	9	30			30	time in register D
	A	57		LDD	A	
	B	7B		LDA	E	Change second digit of time in minutes into
	C	E6		AND	I	ASCII value and store in register E
	D	0F			0F	
	E	F6		ORA	I	
	F	30			30	
08	B 0	5F		LDE	A	
	1	53		LDH	E	Store time in minutes back into
	2	6A		LDL	D	correct space in memory
	3	22		SHLD	HL	
	4	10			10	
	5	30			30	
	6	79		LDA	C	Change first digit of time in hours
	7	0F		RRC		into ASCII value and store in register B
	8	0F		RRC		
	9	0F		RRC		
	A	0F		RRC		
	B	E6		AND	I	
	C	0F			0F	
	D	F6		ORA	I	
	E	30			30	
	F	47		LDB	A	

HEXADECIMAL			MNEMONIC			TITLE DATE
PAGE ADR	LINE ADR	INSTR	LABEL	INSTR	MODIFIER	COMMENTS
08	C 0	79		LDA	C	Change second digit of time in hours
	1	E6		AND	I	into ASCII value and
	2	0F			0F	
	3	F6		ORA	I	
	4	30			30	
	5	4F		LDC	A	store in register C
	6	61		LDH	C	Store time in hours back into original
	7	68		LDL	B	space in memory
	8	22		SHLD	HL	
	9	00			00	
	A	30			30	
08	C B	CD	WAIT-CHECK	JSR	UN(CIN)	Input data from DEC-writer
	C	01			01	
	D	0C			0C	
	E	CD		JSR	UN(COUT)	Echo data back to DEC-writer
	F	10			10	
08	D 0	0C			0C	
	1	E6		AND	I	Set unused MS bit low
	2	7F			7F	
	3	FE		CPA	I	Check for W indicating wait mode
	4	57			57	
	5	C2		JMP	ZO(RUN)	If no W jump to run routine
	6	E5			E5	
	7	08			08	
	8	C1		PLP	BC	Pull top two values off stack
	9	D1		PLP	DE	
	A	F5		PSP	AF	Push W onto stack
	B	C5		PSP	BC	Push direction of grain indicator back onto stack
	C	21		LPI	HL	Load starting address of message
	D	52			52	stating that system is in wait mode
	E	15			15	
	F	CD		JSR	UN(BUFOUT)	Output message to DEC-writer

HEXADECIMAL			MNEMONIC			TITLE DATE
PAGE ADR	LINE ADR	INSTR	LABEL	INSTR	MODIFIER	COMMENTS
08	E 0	10			10	
	1	0C			0C	
	2	C3		JMP	UN(WAIT-CHECK)	Jump to WAIT-CHECK routine to check
	3	CB			CB	for any new inputs
	4	08			08	
08	E 5	FE	RUN	CPA	I	Check for R, indicating run mode
	6	52			52	
	7	C2		JMP	ZO(WAIT-CHECK)	If not R go back to routine to wait for
	8	CB			CB	new inputs
	9	08			08	
	A	C1		PLP	BC	Pull top two values off stack
	B	D1		PLP	DE	
	C	F5		PSP	AF	Push R onto stack
	D	C5		PSP	BC	Push direction of grain indicator onto stack
	E	21		LPI	HL	Load starting address for message
	F	60			60	stating system is in RUN mode
08	F 0	15			15	
	1	CD		JSR	UN(BUFOUT)	Output message to DEC-Writer
	2	10			10	
	3	0C			0C	
	4	C3		JMP	UN(WAIT-CHECK)	Jump to routine to wait for
	5	CB			CB	new inputs
	6	08			08	
	7					
	8					
	9					
	A					
	B					
	C					
	D					
	E					
	F					

HEXADECIMAL PAGE ADR	LINE ADR	INSTR	MNEMONIC LABEL	INSTR	MODIFIER	TITLE / DATE COMMENTS
0C	0 0	00		NOP		Subroutine to input character from DEC-writer
0C	0 1	DB	CIN	INP		Check DEC writer status
	2	21			21	
	3	0F		RRC		
	4	0F		RRC		
	5	D2		JMP	CO(CIN)	Cycle until data is received
	6	01			01	
	7	0C			0C	
	8	DB		INP		Input data
	9	20			20	
	A	E6		AND	I	Set parity bit low
	B	7F			7F	
	C	FE		CPA	I	Check for rubout and set flags
	D	7F			7F	
	E	C9		RET	UN	Return to calling routine
	F	00		NOP		Subroutine to output character to DEC-writer
0C	1 0	F5	COUT	PSP	AF	Save Register A and flags
0C	1 1	DB	COUT 1	INP		Check DEC-writer
	2	21			21	
	3	0F		RRC		
	4	D2		JMP	CO(COUT 1)	Cycle until DEC-writer is ready to receive data
	5	11			11	
	6	0C			0C	
	7	F1		PLP	AF	Get data stored in Register A
	8	D3		OUT		Send data to DEC-writer status
	9	20			20	
	A	C9		RET	UN	Return to calling routine
	B	00		NOP		Subroutine to output whatever is in memory
	C	00		NOP		until 00 i[illegible] encountered
0C	1 D	F5	BUFOUT	PSP	AF	Save Register A and flags
0C	1 E	7E	BUFOUT 1	LDA	M	Get data to be output to DEC-writer
	F	23		INCP	HL	

HEXADECIMAL PAGE ADR	LINE ADR	INSTR	MNEMONIC LABEL	INSTR	MODIFIER	TITLE / DATE COMMENTS
0C	2 0	B7		ORA	A	Check data for being zero
	1	C4		JSR	ZD (COUT)	If data is non-zero output to DEC-writer
	2	10			10	
	3	0C			0C	
	4	C2		JMP	ZD (BUFOUT 1)	If data is non-zero keep sending data
	5	1E			1E	to DEC-writer
	6	0C			0C	
	7	F1		PLP	AF	Recall Register A and flags
	8	C9		RET	UN	Return to calling routine
	9	00		NOP		Subroutine to input data from DEC-writer
	A	00		NOP		and store in memory in ASCII form
0C	2 B	CD	READ-DEC-ASCII	JSR	UN(CIN)	Read data from DEC-writer
	C	01			01	
	D	0C			0C	
	E	CD		JSR	UN(COUT)	Echo data back to DEC-writer
	F	10			10	
0C	3 0	0C			0C	
	1	E6		AND	I	Set parity bit low
	2	7F			7F	
	3	FE		CPA	I	Check for Carriage Return
	4	0D			0D	
	5	C8		RET	ZI	If CR return to calling routine
	6	77		STAN	HL	Store in memory and increment memory address
	7	23		INCP	HL	
	8	C3		JMP	UN(READ-DEC-ASCII)	Jump to beginning of subroutine to
	9	2B			2B	Continue data input
	A	0C			0C	
	B	00		NOP		Subroutine to read data from DEC-writer
	C	00		NOP		and store in memory in Actual form
0C	3 D	CD	READ-DEC-ACT	JSR	UN(CIN)	Read data from DEC-writer
	E	01			01	
	F	0C			0C	

HEXADECIMAL PAGE ADR	LINE ADR	INSTR	MNEMONIC LABEL	INSTR	MODIFIER	TITLE DATE COMMENTS
0C	4 0	CD		JSR	UN(COUT)	Echo value back to DEC-writer
	1	10			10	
	2	0C			0C	
	3	FE		CPA	I	Check for carriage return
	4	0D			0D	
	5	C8		RET	Z1	If CR return to calling routine
	6	FE		CPA	I	Check for decimal
	7	2E			2E	
	8	CA		JMP	Z1 (READ-DEC-ACT)	If decimal, jump to READ-DEC-ACT without
	9	30			30	storing decimal data
	A	0C			0C	
	B	E6		AND	I	Set 4 MS bits low to convert ASCII
	C	0F			0F	number into actual number
	D	77		STAN	HL	Store in memory and increment memory address
	E	23		INCP	HL	
	F	C3		JMP	UN(READ-DEC-ACT)	Jump to READ-DEC-ACT to input more data
0C	5 0	30			30	
	1	0C			0C	
	2	00		NOP		Subroutine to read data from moisture meter
	3	00		NOP		and store in memory in Actual form
0C	5 4	DB	READ-MC-ACT	INP		Input first digit of moisture reading
	5	01			01	which is on 4 LS bits of port 01
	6	E6		AND	I	Set 4 MS bits of data low to convert
	7	0F			0F	data to actual form
	8	77		STAN	HL	Store in memory and increment memory address
	9	23		INCP	HL	
	A	DB		INP		Input second and third digits of
	B	30			30	moisture reading
	C	47		LDB	A	
	D	E6		AND	I	Convert second digit moisture reading
	E	F0			F0	into actual form
	F	0F		RRC		

HEXADECIMAL PAGE ADR	LINE ADR	INSTR	MNEMONIC LABEL	INSTR	MODIFIER	TITLE DATE COMMENTS
0C	6 0	0F		RRC		
	1	0F		RRC		
	2	0F		RRC		
	3	77		STAN	HL	Store in memory and increment memory address
	4	23		INCP	HL	
	5	78		LDA	B	Convert third digit of moisture reading
	6	E6		AND	I	into actual form
	7	0F			0F	
	8	77		STAN	HL	Store in memory
	9	C9		RET	UN	Return to calling routine
	A	00		NOP		Subroutine to read counter value which can be one or two
	B	00		NOP		bytes from DEC-writer & store as one decimal byte value in memor
0C	6 C	CD	READ-DEC-COUNT	JSR	UN(CIN)	Read data from DEC-writer
	D	01			01	
	E	0C			0C	
	F	CD		JSR	UN(COUT)	Echo data back to DEC-writer
0C	7 0	10			10	
	1	0C			0C	
	2	E6		AND	I	Change data from ASCII to actual form
	3	0F			0F	and store in register B
	4	47		LDB	A	
	5	CD		JSR	UN(CIN)	Read data from DEC-writer
	6	01			01	
	7	0C			0C	
	8	CD		JSR	UN(COUT)	Echo data back to DEC-writer
	9	10			10	
	A	0C			0C	
	B	E6		AND	I	Set parity bit low
	C	7F			7F	
	D	FE		CPA	I	Check for carriage return
	E	0D			0D	
	F	C2		JMP	ZO (2-DIGIT)	If not CR it is second byte of data so jump

HEXADECIMAL PAGE ADR	LINE ADR	INSTR	LABEL	MNEMONIC INSTR	MODIFIER	TITLE / COMMENTS	DATE
0C	8 0	85			85	to routine to handle 2nd data byte	
	1	0C			0C		
	2	78		LDA	B	If CR store in memory	
	3	77		STAN	HL		
	4	C9		RET	UN	return to calling routine	
0C	8 5	E6	2-DIGIT	AND	I	Change data from ASCII to actual value	
	6	0F			0F	and store in register C	
	7	4F		LDC	A		
	8	78		LDA	B	Load first byte of data from register B and	
	9	07		RLC		rotate 4 LS bits into 4 MS bits	
	A	07		RLC			
	B	07		RLC			
	C	07		RLC			
	D	B1		ORA	C	Combine both bytes together and store in	
	E	77		STAN	HL	memory as a decimal value	
0C	8 F	CD	REPEAT	JSR	UN(CIN)	Read data from DEC-Writer	
0C	9 0	01			01		
	1	0C			0C		
	2	E6		AND	I	Set parity bit low	
	3	7F			7F		
	4	FE		CPA	I	Check for carriage return	
	5	0D			0D		
	6	C2		JMP	ZD(REPEAT)	If not CR, cycle until CR is read	
	7	8F			8F		
	8	0C			0C		
	9	CD		JSR	UN(COUT)	Echo CR back to DEC-Writer	
	A	10			10		
	B	0C			0C		
	C	C9		RET	UN	return to calling routine	
	D	00		NOP		Subroutine to reset calculator for a new calculator	
0C	9 E	3E	RESET	LDA	I	Send MS bit low on port 02 to	
	F	7F			7F	signal a reset to calculator	
0C	A 0	D3		OUT			
	1	02			02		
	2	00		NOP		Reset line must be held low for at least	
	3	00		NOP		25 . sec	
	4	00		NOP			
	5	00		NOP			
	6	00		NOP			
	7	00		NOP			
	8	3E		LDA	I	Send all lines high on output port 02	
	9	FF			FF	to release reset	
	A	D3		OUT			
	B	02			02		
	C	C9		RET	UN	return to calling routine	
	D	00		NOP		Load instruction code into calculator, length of	
	E	00		NOP		instruction code set by D register counter	
0C	A F	DB	LOAD-W-COUNT	INP		Read calculator	
0C	B 0	03			03		
	1	17		RAL		Check READY line.	
	2	D2		JMP	CD(LOAD-W-COUNT)	Cycle until READY line goes high	
	3	AF			AF		
	4	0C			0C		
	5	0A		LDAN	BC	Load data stored in memory address by BC	
	6	03		INCP	BC	and increment to next memory address	
	7	F6		ORA	I	Set STROBE-IN line high and output	
	8	40			40	it with data to calculator	
	9	D3		OUT			
	A	03			03		
	B	3E		LDA	I	Clear STROBE-IN line	
	C	00			00		
	D	D3		OUT			
	E	03			03		
	F	15		DEC	D	Decrement register D	

HEXADECIMAL PAGE ADR	LINE ADR	INSTR	MNEMONIC LABEL	INSTR	MODIFIER	TITLE DATE COMMENTS
0C	C 0	C8		RET	Z1	If D=0 return to calling routine
	1	C3		JMP	UN(LOAD-W-COUNT)	Otherwise repeat cycle
	2	3F			3F	
	3	0C			0C	
	4	00		NOP		
	5	00		NOP		
	6	00		NOP		
	7	00		NOP		Subroutine to instruct calculator that it is to be read
0C	C 8	DB	OUT	INP		Read calculator status
	9	03			03	
	A	17		RAL		Check if READY line is high
	B	D2		JMP	C0(OUT)	Cycle until READY line goes high
	C	C8			C8	
	D	0C			0C	
	E	3E		LDA	I	Load accumulator with OUT instruction
	F	16			16	
0C	D 0	F6		ORA	I	Send STROBE-IN line high
	1	40			40	
	2	D3		OUT		Output to calculator
	3	03			03	
	4	3E		LDA	I	Clear STROBE-IN line
	5	00			00	
	6	D3		OUT		
	7	03			03	
0C	D 8	DB	OUT 2	INP		Read calculator status
	9	03			03	
	A	17		RAL		Check if READY line is high
	B	D2		JMP	C0 (OUT 2)	Cycle until READY line goes high
	C	D8			D8	
	D	0C			0C	
	E	3E		LDA	I	Load accumulator with second part of OUT
	F	11			11	instruction, can be any value, 11 is used

HEXADECIMAL PAGE ADR	LINE ADR	INSTR	MNEMONIC LABEL	INSTR	MODIFIER	TITLE DATE COMMENTS
0C	E 0	F6		ORA	I	Send STROBE-IN line high
	1	40			40	
	2	D3		OUT		Output to calculator
	3	03			03	
	4	3E		LDA	I	Clear STROBE-IN line
	5	00			00	
	6	D3		OUT		
	7	03			03	
	8	C9		RET	IN	return to calling routine
	9	00		NOP		Subroutine to read answer from calculator
0C	E A	48	READ-CAL	LDC	B	Save register B in register C
0C	E B	DB	READ-CAL 1	INP		Check DATA AVAILABLE* line
	C	03			03	
	D	17		RAL		
	E	17		RAL		
	F	DA		JMP	C1 (READ-CAL 1)	Cycle until DATA AVAILABLE* line goes low
0C	F 0	EB			EB	
	1	0C			0C	
	2	DB	DATA-READ	INP		Input data from calculator
	3	03			03	
	4	E6		AND	I	Set 4 MS bits to zero
	5	0F			0F	
	6	12		STAX	DE	Store in memory space addressed by register pair DE
	7	13		INCP	DE	and increment to next memory address
	8	3E		LDA	I	Set STROBE-OUT high
	9	80			80	
	A	D3		OUT		output to calculator
	B	03			03	
	C	3E		LDA	I	Clear STROBE-OUT to be able to read more data
	D	00			00	
	E	D3		OUT		output to calculator
	F	03			03	

ORIGINAL PAGE IS
OF POOR QUALITY

HEXADECIMAL			MNEMONIC			TITLE DATE
PAGE ADR	LINE ADR	INSTR	LABEL	INSTR	MODIFIER	COMMENTS
00	0 0	05		DEC	B	Decrement B
	1	C2		JMP	ZO (READ-CAL 1)	If B is non-zero read more data
	2	EB			EB	
	3	0C			0C	
	4	79		LDA	C	Check number of digits that were read
	5	FE		CPA	I	from calculator, If anything but 05 bytes
	6	05			05	were read, it doesn't need rounding
	7	C0		RET	ZO	so return to calling routine.
00	0 8	DB	DATA 1	INP		Check DATA AVAILABLE* line
	9	03			03	
	A	17		RAL		
	B	17		RAL		
	C	DA		JMP	C1(DATA 1)	Cycle until DATA AVAILABLE* line goes low
	D	08			08	
	E	00			00	
00	0 F	DB	DATA-READ 1	INP		Input data from calculator card
00	1 0	03			03	
	1	E6		AND	I	Set 4 MS bits to zero
	2	0F			0F	
	3	FE		CPA	I	
	4	05			05	
	5	DA		JMP	CI(CONTINUE 1)	If value is less than 5 it doesn't need
	6	42			42	rounding so jump to CONTINUE 1
	7	00			00	
	8	1B		DECP	DE	Load third digit, increment once and
	9	1A		LDAN	DE	store back into memory
	A	3C		INC	A	
	B	12		STAN	DE	
	C	FE		CPA	I	Jump to CONTINUE 1 if value
	D	CA			0A	stored was a decimal value
	E	C2		JMP	ZO(CONTINUE 1)	
	F	42			42	

HEXADECIMAL			MNEMONIC			TITLE DATE
PAGE ADR	LINE ADR	INSTR	LABEL	INSTR	MODIFIER	COMMENTS
00	2 0	00			00	
	1	27		DAA		If value was 0A, convert to decimal
	2	E6		AND	I	value, and store back into memory
	3	0F			0F	
	4	12		STAN	DE	
	5	1B		DECP	DE	Load second digit, increment once, and
	6	1A		LDAN	DE	store back into memory
	7	3C		INC	A	
	8	12		STAN	DE	
	9	FE		CPA	I	Jump to CONTINUE 1 if value stored
	A	0A			0A	was a decimal value
	B	C2		JMP	ZO (CONTINUE 1)	
	C	42			42	
	D	00			00	
	E	27		DAA		If value was 0A, convert to decimal value,
	F	E6		AND	I	and store back into memory.
00	3 0	0F			0F	
	1	12		STAN	DE	
	2	1B		DECP	DE	Load first digit, increment once, and
	3	1A		LDAN	DE	store back into memory
	4	3C		INC	A	
	5	12		STAN	DE	
	6	FE		CPA	I	Jump to CONTINUE 1 if value stored
	7	0A			0A	was a decimal value
	8	C2		JMP	ZO (CONTINUE 1)	
	9	42			42	
	A	00			00	
	B	3E		LDA	I	If first digit was 0A, change to 01
	C	01			01	and load back into memory
	D	12		STAN	DE	
	E	1B		DECP	DE	Decrement decimal point position indicator
	F	1A		LDAN	DE	once and store back into memory.

HEXADECIMAL PAGE ADR	LINE ADR	INSTR	MNEMONIC LABEL	INSTR	MODIFIER	TITLE DATE COMMENTS
00	40	3D		DEC	A	
	1	12		STAN	DE	
00	42	3E	CONTINUE 1	LDA	I	Set STROBE-OUT line high
	3	80			80	
	4	D3		OUT		and output to calculator
	5	03			03	
	6	3E		LDA	I	Clear STROBE-OUT line so more data can be read
	7	00			00	
	8	D3		OUT		and output to calculator
	9	03			03	
	A	C9		RET	UN	Return to calling routine
	B	00		NOP		Subroutine for copying values from one memory to another
	C	00		NOP		memory block, Counter in Register B for number of bytes copied
00	4D	7E	COPY	LDAN	HL	Load value from original memory space and
	E	23		INCP	HL	
	F	12		STAN	DE	then store it in desired memory space
00	50	13		INCP	DE	
	1	05		DEC	B	Decrement counter
	2	C8		RET	ZI	If counter is zero return to calling routine
	3	C3		JMP	UN(COPY)	If non-zero keep copying data
	4	4D			4D	
	5	00			00	
	6	00		NOP		Subroutine to convert values from ASCII form into calculator form
00	57	7E	ASCII-CAL	LDAN	HL	Load value from memory and increment to next
	8	23		INCP	HL	memory address
	9	FE		CPA	I	Check for ASCII decimal
	A	2E			2E	
	B	C2		JMP	ZO (NUMBER)	If not decimal jump to routine to convert
	C	67			67	ASCII numbers into calculator numbers
	D	00			00	
	E	3E		LDA	I	If ASCII decimal, load accumulator with
	F	0A			0A	Calculator decimal and store in calculator memory space

HEXADECIMAL PAGE ADR	LINE ADR	INSTR	MNEMONIC LABEL	INSTR	MODIFIER	TITLE DATE COMMENTS
00	60	12		STAN	DE	
	1	13		INCP	DE	
	2	05		DEC	B	Decrement counter
	3	C8		RET	ZI	If counter is zero return to calling routine
	4	C3		JMP	UN(ASCII-CAL)	If non-zero continue converting data
	5	57			57	
	6	00			00	
00	67	E6	NUMBER	AND	I	Convert ASCII number into Calculator number
	8	0F			0F	
	9	12		STAN	DE	and store in Calculator memory space
	A	13		INCP	DE	
	B	05		DEC	B	Decrement counter
	C	C8		RET	ZI	If counter is zero return to calling routine
	D	C3		JMP	UN(ASCII-CAL)	If non-zero continue converting data
	E	57			57	
	F	00			00	
00	70	00		NOP		Subroutine for converting answer from Calculator
	1	00		NOP		into calculator form for future calculations
00	72	7E	ANS-CAL	LDAN	HL	Increment past byte showing sign
	3	23		INCP	HL	
	4	05		DEC	B	and decrement counter
	5	7E		LDAN	HL	Load value of decimal point position indicator
	6	23		INCP	HL	
00	77	FE	0B	CPA	I	Check decimal point position value for being 0B
	8	0B			0B	
	9	C2		JMP	ZO (0A)	If not 0B jump to check 0A
	A	7E			7E	
	B	00			00	
	C	0E		LDC	I	If 0B load decimal counter with 01
	D	01			01	
00	7E	FE	0A	CPA	I	Check decimal point position value for being 0A
	F	0A			0A	

HEXADECIMAL			MNEMONIC			TITLE DATE
PAGE ADR	LINE ADR	INSTR	LABEL	INSTR	MODIFIER	COMMENTS
00	8 0	C2		JMP	ZO (09)	If not 0A, jump to check 09
	1	85			85	
	2	00			00	
	3	0E		LDC	I	If 0A, load decimal counter with 02
	4	02			02	
00	8 5	FE	09	CPA	I	Check decimal point position value for being 09
	6	09			09	
	7	C2		JMP	ZO (08)	If not 09, jump to check 08
	8	8C			8C	
	9	00			00	
	A	0E		LDC	I	If 09, load decimal counter with 03
	B	03			03	
00	8 C	FE	08	CPA	I	Check decimal point position value for being 08
	D	08			08	
	E	C2		JMP	ZO (07)	If not 08, jump to check 07
	F	93			93	
00	9 0	00			00	
	1	0E		LDC	I	If 08, load decimal counter with 04
	2	04			04	
00	9 3	FE	07	CPA	I	Check decimal point position value for being 07
	4	07			07	
	5	C2		JMP	ZO (06)	If not 07, jump to check 06
	6	9A			9A	
	7	00			00	
	8	0E		LDC	I	If 07, load decimal counter with 05
	9	05			05	
00	9 A	FE	06	CPA	I	Check decimal point position value for being 06
	B	06			06	
	C	C2		JMP	ZO (05)	If not 06, jump to check 05
	D	A1			A1	
	E	00			00	
	F	0E		LDC	I	If 06, load decimal counter with 06

HEXADECIMAL			MNEMONIC			TITLE DATE
PAGE ADR	LINE ADR	INSTR	LABEL	INSTR	MODIFIER	COMMENTS
00	A 0	06			06	
00	A 1	FE	05	CPA	I	Check decimal point position value for being 05
	2	05			05	
	3	C2		JMP	ZO (04)	If not 05, jump to check 04
	4	A8			A8	
	5	00			00	
	6	0E		LDC	I	If 05, load decimal counter with 07
	7	07			07	
00	A 8	FE	04	CPA	I	Check decimal point position value for being 04
	9	04			04	
	A	C2		JMP	ZO (NUMBER 1)	If not 04, correct decimal counter value has
	B	AF			AF	already been loaded so jump to NUMBER
	C	00			00	
	D	0E		LDC	I	If 04, load decimal counter with 08
	E	08			08	
00	A F	7E	NUMBER 1	LDAN	HL	Load value from answer memory space
00	B 0	23		INCP	HL	
	1	12		STAN	DE	Store in Calculator memory space since number
	2	13		INCP	DE	is already in correct form
	3	0D		DEC	C	Decrement decimal counter
	4	C2		JMP	ZO (CHECK-B)	If decimal counter not zero jump to check
	5	BC			BC	digit counter
	6	00			00	
	7	3E		LDA	I	If decimal counter is zero load calculator decimal
	8	0A			0A	
	9	12		STAN	DE	Store in calculator memory space
	A	13		INCP	DE	
	B	05		DEC	B	Decrement digit counter for decimal
00	B C	05	CHECK-B	DEC	B	Decrement digit counter for number value
	D	C8		RET	ZI	return to calling routine if digit counter is zero
	E	C3		JMP	UN(NUMBER 1)	If digit counter non-zero convert more data.
	F	AF			AF	

HEXADECIMAL			MNEMONIC			TITLE / DATE
PAGE ADR	LINE ADR	INSTR	LABEL	INSTR	MODIFIER	COMMENTS
00	C 0	00			00	
	1	00		NOP		Subroutine to convert a three digit Actual number
	2	00		NOP		into a two digit, decimal, digit ASCII number
00	C 3	7E	ACT-ASCII	LDAN	HL	Load actual form of first digit
	4	23		INCP	HL	
	5	F6		ORA -	I	change to ASCII form
	6	30			30	
	7	12		STAN	DE	Store in ASCII memory space
	8	13		INCP	DE	
	9	7E		LDAN	HL	Load actual form of second digit
	A	23		INCP	HL	
	B	F6		ORA	I	Change to ASCII form
	C	30			30	
	D	12		STAN	DE	Store in ASCII memory space
	E	13		INCP	DE	
	F	3E		LDA	I	Load ASCII decimal point
00	D 0	2E			2E	
	1	12		STAN	DE	Store in ASCII memory space
	2	13		INCP	DE	
	3	7E		LDAN	HL	Load actual form of third digit
	4	23		INCP	HL	
	5	F6		ORA	I	Change to ASCII form
	6	30			30	
	7	12		STAN	DE	Store in ASCII memory space
	8	13		INCP	DE	
	9	C9		RET	UN	Return to calling routine
	A	00		NOP		Convert three digit actual value into a
	B	00		NOP		two digit, decimal, one digit Calculator value
00	D C	7E	ACT-CAL	LDAN	HL	Load actual form of first digit which is
	D	23		INCP	HL	same as Calculator form
	E	12		STAN	DE	Store into calculator form memory space
	F	13		INCP	DE	

HEXADECIMAL			MNEMONIC			TITLE / DATE
PAGE ADR	LINE ADR	INSTR	LABEL	INSTR	MODIFIER	COMMENTS
00	E 0	7E		LDAN	HL	Load actual form of second digit
	1	23		INCP	HL	
	2	12		STAN	DE	Store into calculator form memory space
	3	13		INCP	DE	
	4	3E		LDA	I	Load calculator decimal point
	5	0A			0A	
	6	12		STAN	DE	Store in calculator form memory space
	7	13		INCP	DE	
	8	7E		LDAN	HL	Load actual form of third digit
	9	23		INCP	HL	
	A	12		STAN	DE	Store into calculator form memory space
	B	13		INCP	DE	
	C	C9		RET	UN	Return to calling routine
	D	00		NOP		Subroutine to convert data from calculator form
	E	00		NOP		into ASCII form with Register B as a digit counter
	F	00		NOP		
00	F 0	7E	CAL-ASCII	LDAN	HL	Load calculator byte
	1	23		INCP	HL	
	2	FE		CPA	I	Check for calculator decimal point
	3	0A			0A	
	4	C2		JMP	Z0 (NUMBER 2)	If not decimal jump to NUMBER 2
	5	00			00	
	6	0E			0E	
	7	3E		LDA	I	If decimal point, load ASCII decimal point
	8	2E			2E	
	9	12		STAN	DE	Store in ASCII form memory space
	A	13		INCP	DE	
	B	05		DEC	B	Decrement counter
	C	C8		RET	Z1	If counter is zero return to calling routine
	D	C3		JMP	UN(CAL-ASCII)	Otherwise continue converting data
	E	F0			F0	
	F	0D			0D	

HEXADECIMAL			MNEMONIC			TITLE DATE
PAGE ADR	LINE ADR	INSTR	LABEL	INSTR	MODIFIER	COMMENTS
0E	0 0	F6	NUMBER 2	ORA	I	Convert Calculator number into ASCII number
	1	30			30	
	2	12		STAN	DE	Store in ASCII form memory space
	3	13		INCP	DE	
	4	05		DEC	B	Decrement counter
	5	C8		RET	Z1	If counter is zero return to calling routine
	6	C3		JMP	UN(CAL-ASCII)	Otherwise continue converting data
	7	F0			F0	
	8	00			00	
	9	00		NOP		Subroutine to convert Calculator form data
	A	00		NOP		into actual form with Register B as a digit counter
0E	0 B	7E	CAL-ACT	LDAN	HL	Load calculator byte
	C	23		INCP	HL	
	D	FE		CPA	I	Check for Calculator decimal point
	E	0A			0A	
	F	C2		JMP	Z0(NUMBER 3)	If not decimal jump to NUMBER 3
0E	1 0	17			17	
	1	0E			0E	
	2	05		DEC	B	Decrement counter
	3	C8		RET	Z1	If counter is zero return to calling routine
	4	C3		JMP	UN(CAL-ACT)	Otherwise coutinue converting data after
	5	0B			0B	omitting decimal point
	6	0E			0E	
0E	1 7	12	NUMBER 3	STAN	DE	Store number into actual form memory space
	8	13		INCP	DE	
	9	05		DEC	B	Decrement counter
	A	C8		RET	Z1	If counter is zero return to calling routine
	B	C3		JMP	UN(CAL-ACT)	Otherwise continue converting data
	C	0B			0B	
	D	0E			0E	
	E	00		NOP		Specialized subroutine for taking average MC answer
	F	00		NOP		from calculator which consists of a sign byte, decimal
0E	2 0	00		NOP		point position byte, and three digits, and convert to
	1	00		NOP		Calculator form with two digits, decimal, and a digit
0E	2 2	7E	AVG-ANS-CAL	LDAN	HL	Increment past sign byte
	3	23		INCP	HL	
	4	7E		LDAN	HL	Load decimal point position indicator byte
	5	23		INCP	HL	
	6	FE		CPA	I	Check for 0B decimal point
	7	0B			0B	
	8	CA		JMP	Z1(0B-DIGIT)	If 0B jump to routine to handle 0B decimal point
	9	3C			3C	
	A	0E			0E	
0E	2 B	7E	0A-DIGIT	LDAN	HL	Load first byte of answer form, no conversion required
	C	23		INCP	HL	
	D	12		STAN	DE	Store as first digit of Calculator form avg MC
	E	13		INCP	DE	
	F	7E		LDAN	HL	Load second byte of Answer form
0E	3 0	23		INCP	HL	
	1	12		STAN	DE	Store as second digit of Calculator form avg MC
	2	13		INCP	DE	
	3	3E		LDA	I	Load calculator decimal point
	4	0A			0A	
	5	12		STAN	DE	Store into calculator form memory space
	6	13		INCP	DE	
	7	7E		LDAN	HL	Load third byte of answer form
	8	23		INCP	HL	
	9	12		STAN	DE	Store as third digit of calculator form avg MC
	A	13		INCP	DE	
	B	C9		RET	UN	Return to calling routine
0E	3 C	3E	0B-DIGIT	LDA	I	Load zero
	D	00			00	
	E	12		STAN	DE	Store as first digit of Calculator form avg MC
	F	13		INCP	DE	

HEXADECIMAL			MNEMONIC			TITLE DATE
PAGE ADR	LINE ADR	INSTR	LABEL	INSTR	MODIFIER	COMMENTS
0E	4 0	7E		LDAN	HL	Load first byte of answer form
	1	23		INCP	HL	
	2	12		STAN	DE	Store as second digit of Calculator form avg MC
	3	13		INCP	DE	
	4	3E		LDA	I	Load Calculator decimal point
	5	0A			0A	
	6	12		STAN	DE	Store into Calculator form avg MC space
	7	13		INCP	DE	
	8	7E		LDAN	HL	Load second byte of answer form
	9	23		INCP	HL	
	A	12		STAN	DE	Store as third digit of Calculator form avg MC
	B	13		INCP	DE	
	C	7E		LDAN	HL	Load third byte of answer form
	D	23		INCP	HL	
	E	FE		CPA	I	Check if rounding of last digit is required
	F	05			05	
0E	5 0	D8		RET	C1	Return to calling routine if third byte<05
	1	1B		DECP	DE	Load third digit of Calculator form avg. MC
	2	1A		LDAN	DE	
	3	3C		INC	A	Round digit up one
	4	12		STAN	DE	Store back as third digit of Calculator form avg MC
	5	FE		CPA	I	Check if third digit was non decimal
	6	0A			0A	
	7	C0		RET	Z1	Return to calling routine if third digit is decimal value
	8	27		DAA		If non decimal convert to decimal value
	9	E6		AND	I	
	A	0F			0F	
	B	12		STAN	DE	Store as third digit of Calculator form avg MC
	C	1B		DECP	DE	Decrement past decimal point
	D	1B		DECP	DE	Decrement to second digit
	E	1A		LDAN	DE	Load second digit of Calculator form avg MC
	F	3C		INC	A	Round digit up one

HEXADECIMAL			MNEMONIC			TITLE DATE
PAGE ADR	LINE ADR	INSTR	LABEL	INSTR	MODIFIER	COMMENTS
0E	6 0	12		STAN	DE	Store as second digit of Calculator form avg MC
	1	FE		CPA	I	Check if second digit was non decimal
	2	0A			0A	
	3	C0		RET	ZO	Return to calling routine if second digit was decimal value
	4	27		DAA		If non decimal convert to decimal value
	5	E6		AND	I	
	6	0F			0F	
	7	12		STAN	DE	Store as second digit of Calculator form avg MC
	8	1B		DECP	DE	Decrement to first digit
	9	1A		LDAN	DE	Load first digit
	A	3C		INC	A	Round up one
	B	12		STAN	DE	Store as first digit of Calculator form avg MC
	C	C9		RET	UN	Return to calling routine
	D	00		NOP		Subroutine to convert decimal value into
	E	00		NOP		hexadecimal value, one byte value
0E	6 F	7E	DECIMAL-HEX	LDAN	HL	Load value stored in memory addressed by HL
0E	7 0	47		LDB	A	Save in register B
	1	E6		AND	I	Use 4 MS bits as counter value for number
	2	F0			F0	of times to go thru decrement cycle
	3	0F		RRC		
	4	0F		RRC		
	5	0F		RRC		
	6	0F		RRC		
	7	4F		LDC	A	Store counter in register C
	8	CA		JMP	Z1 (STORE)	If 4 MS bits were zero data doesn't need
	9	85			85	converting so go to routine to store data
	A	0E			0E	
0E	7 B	05	DEC	DEC	B	Decrement original value six times to convert
	C	05		DEC	B	from hexadecimal value to decimal value
	D	05		DEC	B	
	E	05		DEC	B	
	F	05		DEC	B	

HEXADECIMAL			MNEMONIC			TITLE DATE
PAGE ADR	LINE ADR	INSTR	LABEL	INSTR	MODIFIER	COMMENTS
0E	8 0	05		DEC	B	
	1	0D		DEC	C	Decrement counter
	2	C2		JMP	Z0(DEC)	If counter is non zero jump to go thru
	3	7B			7B	six decrements again
	4	0E			0E	
0E	8 5	78	STORE	LDA	B	If counter is zero store converted data
	6	12		STAN	DE	in desired memory space
	7	C9		RET	UN	Return to calling routine
	8	00		NOP		Check if low limit on indiv MC reading is lower
	9	00		NOP		than high limit on indiv MC reading
0E	8 A	21	INDIV-CHECK	LPI	HL	Load starting address where high limit for
	B	50			50	indiv MC reading is stored
	C	30			30	
	D	11		LPI	DE	Load starting address where low limit for
	E	53			53	indiv MC reading is stored
	F	30			30	
0E	9 0	06		LDB	I	Set up counter for number of bytes to be compared
	1	03			03	
0E	9 2	7E	INDIV-CHECK 1	LDAN	HL	Load digit of high limit for indiv MC
	3	23		INCP	HL	reading into register C
	4	4F		LDC	A	
	5	1A		LDAN	DE	Load digit of low limit for indiv MC
	6	13		INCP	DE	reading into accumulator
	7	B9		CPA	C	Compare A with C
	8	DA		JMP	C1(HI-AVG-READ)	If low indiv reading limit digit ≤ high indiv reading/limit
	9	49			49	digit, then low indiv reading limit ≤ high indiv reading
	A	08			08	limit so jump to HI-AVG-READ
	B	C2		JMP	Z0 (READ-LIM-MESS)	If low indiv reading limit digit ≰ high indiv reading
	C	A5			A5	limit, then low indiv reading limit > high indiv
	D	0F			0F	reading limit so jump to READ-MESS
	E	05		DEC	B	Decrement B
	F	CA		JMP	Z1(HI-AVG-READ)	If B=0, all digits have been checked so

HEXADECIMAL			MNEMONIC			TITLE DATE
PAGE ADR	LINE ADR	INSTR	LABEL	INSTR	MODIFIER	COMMENTS
0E	A 0	49			49	jump to HI-AVG-READ
	1	08			08	
	2	C2		JMP	UN(INDIV-CHECK1)	If B≠0 repeat routine
	3	92			92	
	4	0E			0E	
	5	00		NOP		Check if low limit on avg MC is lower than
	6	00		NOP		high limit on avg MC
0E	A 7	21	AVG-CHECK	LPI	HL	Load starting address where high limit for
	8	59			59	avg MC is stored
	9	30			30	
	A	11		LPI	DE	Load starting address where low limit for
	B	5C			5C	avg MC is stored
	C	30			30	
	D	06		LDB	I	Set up counter for number of bytes to be compared
	E	03			03	
0E	A F	7E	AVG-CHECK 1	LDAN	HL	Load digit of high limit for avg MC into
0E	B 0	23		INCP	HL	Register C
	1	4F		LDC	A	
	2	1A		LDAN	DE	Load digit of low limit for avg MC into accumulator
	3	13		INCP	DE	
	4	B9		CPA	C	Compare A with C
	5	DA		JMP	C1(DIRECTION)	If low avg MC limit digit ≤ high avg MC limit digit,
	6	67			67	then low avg MC limit ≤ high avg MC limit so
	7	08			08	jump to DIRECTION in main program
	8	C2		JMP	Z0(AVG-LIM-MESS)	If low avg MC limit digit ≰ high avg MC limit digit
	9	AF			AF	then low avg MC limit > high avg MC limit, so
	A	0F			0F	jump to AVG-MESS
	B	05		DEC	B	Decrement B
	C	CA		JMP	Z1(DIRECTION)	If B = 0, all digits have been checked so
	D	67			67	jump to DIRECTION in main program
	E	C8			08	
	F	C2		JMP	UN(AVG-CHECK 1)	If B≠0, repeat routine

HEXADECIMAL			MNEMONIC			TITLE / DATE
PAGE ADR	LINE ADR	INSTR	LABEL	INSTR	MODIFIER	COMMENTS
0E	C 0	AF			AF	
	1	0E			0E	
	2	00		NOP		Subroutine to check if desired avg MC value
	3	00		NOP		is within avg MC high and low digits
0E	C 4	21	DESIR-AVG-TEST	LPI	HL	Load starting address where desired final avg MC
	5	B3			B3	is stored in Calculator form
	6	30			30	
	7	11		LPI	DE	Load starting address where desired final avg MC
	8	00			00	is to be stored in actual form
	9	31			31	
	A	06		LDB	I	Set up counter for number of byte to be converted
	B	04			04	
	C	CD		JSR	UN(CAL-ACT)	Convert data from Calculator form into
	D	08			08	actual form
	E	0E			0E	
0E	C F	21	HI-DESIR-AVG	LPI	HL	Load starting address where upper limit
0E	D 0	53			53	for avg MC is stored in actual form
	1	30			30	
	2	11		LPI	DE	Load starting address where desired final
	3	00			00	avg MC is stored in actual form
	4	31			31	
	5	06		LDB	I	Set up counter for number of byte to be compared
	6	03			03	
0E	D 7	7E	HI-DESIRED	LDAN	HL	Load byte of upper limit for avg MC into
	8	23		INCP	HL	register C
	9	4F		LDC	A	
	A	1A		LDAN	DE	Load byte of desired final avg MC into accumulator
	B	13		INCP	DE	
	C	B9		CPA	C	Compare A with C
	D	DA		JMP	C1(LO-DESIR-AVG)	If A ≤ C, upper limit for avg MC is okay, so
	E	E7			E7	jump to subroutine to check lower limit for
	F	0E			0E	avg MC
0E	E 0	C2		JMP	Z0 (BAD-DESIR-MESS)	If A ≰ C, then final desired avg MC is higher
	1	B9			B9	than upper limit for avg MC, so jump to
	2	0F			0F	routine to output a message stating this fact
	3	05		DEC	B	Decrement counter
	4	C2		JMP	Z0(HI-DESIRED)	If counter ≠ zero jump to part of routine to check
	5	D7			D7	next byte of data
	6	0E			0E	
0E	E 7	21	LO-DESIR-AVG	LPI	HL	Load starting address where lower limit for
	8	5C			5C	avg MC is stored in actual form
	9	30			30	
	A	11		LPI	DE	Load starting address where desired final
	B	00			00	avg MC is stored in actual form
	C	31			31	
	D	06		LDB	I	Set up counter for number of bytes to be compared
	E	03			03	
0E	E F	7E	LO-DESIRED	LDAN	HL	Load byte of lower limit for avg MC into
0E	F 0	23		INCP	HL	register C
	1	4F		LDC	A	
	2	1A		LDAN	DE	Load byte of desired final avg MC accumulator
	3	13		INCP	DE	
	4	B9		CPA	C	Compare A with C
	5	DA		JMP	C1 (BAD-DESIR-MESS)	If A<C, then desired final avg MC is lower than low
	6	B9			B9	limit for avg MC so jump to subroutine
	7	0F			0F	to output message
	8	C2		JMP	Z0 (READ-INTERVAL)	If A ≮ C, then desired final avg MC is higher than
	9	E8			E8	low limit for avg MC so jump to routine
	A	08			08	to ask next question
	B	05		DEC	B	Decrement counter
	C	CA		JMP	Z1(READ-INTERVAL)	If counter is zero all bytes have been checked, so
	D	E8			E8	jump to routine to ask next question
	E	08			08	
	F	C2		JMP	Z0(LO-DESIRED)	If counter is non-zero jump to part of routine to

HEXADECIMAL PAGE ADR	LINE ADR	INSTR	MNEMONIC LABEL	INSTR	MODIFIER	TITLE / COMMENTS
0F	0 0	EF			EF	check next byte
	1	0E			0E	
	2	00		NOP		Subroutine to test if MC reading is valid, to be
	3	00		NOP		valid it must be ≥ 05.0 and ≤ 50.0
0F	0 4	21	HI-VALID-TEST	LPI	HL	Load starting address where high limit for a
	5	FA			FA	reasonable indiv MC reading is stored in actual form
	6	30			30	
	7	11		LPI	DE	Load starting address where indiv MC reading is
	8	56			56	stored in actual form
	9	30			30	
	A	06		LDB	I	Set up counter for number of bytes to be compared
	B	03			03	
0F	0 C	7E	HI-TEST	LDAN	HL	Load byte of high limit for a valid indiv MC
	D	23		INCP	HL	reading into register C
	E	4F		LDC	A	
	F	1A		LDAN	DE	Load byte of indiv MC reading into accumulator
0F	1 0	13		INCP	DE	
	1	B9		CPA	C	Compare A with C
	2	DA		JMP	C1 (LO-VALID-TEST)	If indiv MC reading byte ≤ valid high limit byte
	3	1F			1F	indiv MC reading ≤ valid high limit so jump to
	4	0F			0F	test against low valid limit
	5	C2		JMP	Z0 (HI-VALID-MESS)	If indiv MC reading byte ≰ valid high limit byte,
	6	C3			C3	indiv MC reading > valid high limit so
	7	0F			0F	jump to output message
	8	05		DEC	B	Decrement counter
	9	CA		JMP	Z1 (LO-VALID-TEST)	If counter equal zero, all bytes have been
	A	1F			1F	checked so jump to test low valid limit
	B	0F			0F	
	C	C3		JMP	UN(HI-TEST)	Otherwise repeat routine
	D	0C			0C	
	E	0F			0F	
0F	1 F	21	LO-VALID-TEST	LPI	HL	Load starting address where low limit for a valid

HEXADECIMAL PAGE ADR	LINE ADR	INSTR	MNEMONIC LABEL	INSTR	MODIFIER	TITLE / COMMENTS
0F	2 0	FD			FD	indiv MC reading is stored in actual form
	1	30			30	
	2	11		LPI	DE	Load starting address where indiv MC reading
	3	56			56	is stored in actual form
	4	30			30	
	5	06		LDB	I	Set up counter for number of bytes to be compared
	6	03			03	
0F	2 7	7E	LO-TEST	LDAN	HL	Load byte of low limit for a valid MC
	8	23		INCP	HL	reading into register C
	9	4F		LDC	A	
	A	1A		LDAN	DE	Load byte of indiv MC reading into accumulator
	B	13		INCP	DE	
	C	B9		CPA	C	Compare A with C
	D	DA		JMP	C1 (LO-VALID-MESS)	If indiv MC reading byte ≤ valid low limit byte
	E	CC			CC	then indiv MC reading ≤ valid low limit so jump
	F	0F			0F	to output message
0F	3 0	C2		JMP	Z0 (SAMPLE-CONT)	If indiv MC reading byte ≰ valid low limit byte
	1	7E			7E	jump back to main program where check ocurred
	2	09			09	
	3	05		DEC	B	Decrement B
	4	CA		JMP	Z1(SAMPLE-CONT)	If counter is zero desired number of bytes have been
	5	7E			7E	compared so jump back to main program where
	6	09			09	check routine ocurred
	7	C3		JMP	UN(LO-TEST)	Otherwise repeat routine
	8	27			27	
	9	0F			0F	
	A	00		NOP		Subroutine to check if indiv MC reading exceeds high
	B	00		NOP		limit for indiv MC reading
0F	3 C	7E	HI-INDIV-TEST	LDAN	HL	Load byte of high limit for indiv MC reading
	D	23		INCP	HL	into Register C
	E	4F		LDC	A	
	F	1A		LDAN	DE	Load byte of indiv MC reading into accumulator

HEXADECIMAL PAGE ADR	LINE ADR	INSTR	MNEMONIC LABEL	INSTR	MODIFIER	TITLE / DATE / COMMENTS
0F	4 0	13		INCP	DE	
	1	B9		CPA	C	Compare A with C
	2	D8		RET	C1	If indiv MC reading byte < high limit byte return to calling rou-
	3	C2		JMP	Z0(HI-INDIV-MESS)	If indiv MC reading byte ≥ high limit byte jump to
	4	E4			E4	output message
	5	0F			0F	
	6	05		DEC	B	Decrement counter
	7	C8		RET	Z1	If counter equal zero return to calling routine
	8	C3		JMP	UN(HI-INDIV-TEST)	Otherwise repeat routine
	9	3C			3C	
	A	0F			0F	
	B	00		NOP		Subroutine to check if indiv MC reading is less than
	C	00		NOP		low limit for indiv MC reading
0F	4 D	7E	LO-INDIV-TEST	LDAN	HL	Load byte of low limit for indiv MC reading
	E	23		INCP	HL	into Register C
	F	4F		LDC	A	
0F	5 0	1A		LDAN	DE	Load byte of indiv MC reading into accumulator
	1	13		INCP	DE	
	2	B9		CPA	C	Compare A with C
	3	DA		JMP	C1(LO-INDIV-MESS)	If indiv MC reading byte < low limit for indiv MC reading
	4	EB			EB	byte then indiv MC reading < low limit for indiv MC reading
	5	0F			0F	so jump to output message
	6	C0		RET	Z0	If indiv MC reading byte ≥ low limit byte return to calling routi
	7	05		DEC	B	Decrement counter
	8	C8		RET	Z1	If counter is zero return to calling routine
	9	C3		JMP	UN(LO-INDIV-TEST)	Otherwise repeat routine
	A	4D			4D	
	B	0F			0F	
	C	00		NOP		Subroutine to check, if avg MC exceeds high limit for
	D	00		NOP		avg MC
0F	5 E	7E	HI-AVG-TEST	LDAN	HL	Load byte of high limit for avg MC into Register C
	F	23		INCP	HL	

HEXADECIMAL PAGE ADR	LINE ADR	INSTR	MNEMONIC LABEL	INSTR	MODIFIER	TITLE / DATE / COMMENTS
0F	6 0	4F		LDC	A	
	1	1A		LDAN	DE	Load byte of avg MC into accumulator
	2	13		INCP	DE	
	3	B9		CPA	C	Compare A with C
	4	D8		RET	C1	If avg MC byte < high limit byte return to calling routine
	5	C2		JMP	Z0 (HI-AVG-MESS)	If avg MC byte ≥ high limit byte jump to
	6	F2			F2	output message
	7	0F			0F	
	8	05		DEC	B	Decrement counter
	9	C8		RET	Z1	If counter is zero return to calling routine
	A	C3		JMP	UN(HI-AVG-TEST)	Otherwise repeat routine
	B	5E			5E	
	C	0F			0F	
	D	00		NOP		Subroutine to check if avg MC is less then
	E	00		NOP		low limit for avg MC
0F	6 F	7E	LO-AVG-TEST	LDAN	HL	Load byte of low limit for avg MC into Register C
0F	7 0	23		INCP	HL	
	1	4F		LDC	A	
	2	1A		LDAN	DE	Load byte avg MC into accumulator
	3	13		INCP	DE	
	4	B9		CPA	C	Compare A with C
	5	DA		JMP	C1 (LO-AVG-MESS)	If avg MC byte < low limit for avg MC byte, then
	6	F9			F9	avg MC < low limit for avg MC so jump to output message
	7	0F			0F	
	8	C0		RET	Z0	If avg MC byte ≥ low limit byte return to calling routine
	9	05		DEC	B	Decrement counter
	A	C8		RET	Z1	If counter is zero return to calling routine
	B	C3		JMP	UN(LO-AVG-TEST)	Otherwise repeat routine
	C	6F			6F	
	D	0F			0F	
	E	00		NOP		Subroutine to check if predicted number of
	F	00		NOP		readings have been taken

Hexadecimal: Page Adr	Line Adr	Instr	Mnemonic: Label	Instr	Modifier	Title / Date / Comments
0F	8 0	21	CHECK-READING	LPI	HL	Load starting address when number of readings
	1	E8			E8	remaining is stored in Calculator form
	2	30			30	
	3	06		LDB	I	Set up counter for number of bytes to check
	4	05			05	
0F	8 5	7E	LOAD-ND	LDAN	HL	Load byte of data
	6	FE		CPA	I	Check for Calculator decimal point
	7	0A			0A	
	8	C2		JMP	ZO (ZERO)	If not Calculator decimal point jump to check
	9	8D			8D	data for being zero
	A	0F			0F	
	B	3E		LDA	I	If 0A, change to 00
	C	00			00	
0F	8 D	FE	ZERO	CPA	I	Check for byte being 00
	E	00			00	
	F	C2		JMP	ZO (DESIRED-MC)	If not 00 then all readings haven't been taken
0F	9 0	B8			B8	so jump back to main program to calculate
	1	0A			0A	what remaining MC content should be
	2	23		INCP	HL	Go to next byte
	3	05		DEC	B	Decrement counter
	4	C2		JMP	ZO (LOAD-ND)	If counter is non-zero load another byte to
	5	85			85	be checked
	6	0F			0F	
	7	F1		PLP	AF	If all bytes are 00 pull direction of grain indicator (N) off
	8	3E		LDA	I	stack
	9	59			59	Push new direction of grain indicator (Y) onto stack to
	A	F5		PSP	AF	indicate no more calculations for remaining MC are to be made
	B	21		LPI	HL	Load starting address for message stating all
	C	00			00	predicted readings have been taken
	D	14			14	
	E	CD		JSR	UN(BUFOUT)	Output message to DEC-writer
	F	1D			1D	

Hexadecimal: Page Adr	Line Adr	Instr	Mnemonic: Label	Instr	Modifier	Title / Date / Comments
0F	A 0	0C			0C	
	1	C3		JMP	UN(UNLOAD-GRAIN)	Jump to routine to unload grain from test cell
	2	01			01	
	3	08			08	
	4	00		NOP		Message stating low indiv MC reading limit > high indiv MC read-
0F	A 5	21	READ-LIM-MESS	LPI	HL	Load starting address for message stating low indiv. ing limit
	6	CC			CC	MC reading limit > high indiv MC reading limit
	7	14			14	
	8	CD		JSR	UN(BUFOUT)	Output message to DEC-writer
	9	1D			1D	
	A	0C			0C	
	B	C3		JMP	UN(HI-INDIV-READ)	Jump to routine in main program to repeat
	C	2E			2E	questions about indiv MC reading limits
	D	08			08	
	E	00		NOP		Message stating low avg MC limit > high avg MC limit
0F	A F	21	AVG-LIM-MESS	LPI	HL	Load starting address for message stating low
0F	B 0	19			19	avg MC limit > high avg MC limit
	1	15			15	
	2	CD		JSR	UN(BUFOUT)	Output message to DEC-writer
	3	1D			1D	
	4	0C			0C	
	5	C3		JMP	UN(HI-AVG-READ)	Jump to routine in main program to repeat
	6	49			49	questions about avg MC limits
	7	08			08	
	8	00		NOP		Message stating desired final avg MC is outside avg MC limits
0F	B 9	21	BAD-DESIR-MESS	LPI	HL	Load starting address for message stating desired final
	A	94			94	avg MC is outside limits set for avg MC
	B	14			14	
	C	CD		JSR	UN(BUFOUT)	Output message to DEC-writer
	D	1D			1D	
	E	0C			0C	
	F	C3		JMP	UN(DESIR-FIN-AVG)	Jump to routine in main program to ask question

HEXADECIMAL PAGE ADR	LINE ADR	INSTR	MNEMONIC LABEL	INSTR	MODIFIER	TITLE / DATE COMMENTS
0F	C 0	CE			CE	What is the desired final avg MC
	1	0B			0B	
	2	00		NOP		Message stating MC values relationship to their limits
0F	C 3	21	HI-VALID-MESS	LPI	HL	Load starting address for message stating that an
	4	45			45	excessively high reading has been taken and will be ignored
	5	14			14	
	6	CD		JSR	UN(BUFOUT)	Output message to DEC-writer
	7	10			10	
	8	0C			0C	
	9	C3		JMP	UN(VALUE)	Jump to routine to output value that was read
	A	D2			D2	
	B	0F			0F	
0F	C C	21	LO-VALID-MESS	LPI	HL	Load starting address for message stating that an
	D	6D			6D	excessively low reading has been taken and will be ignored
	E	14			14	
	F	CD		JSR	UN(BUFOUT)	Output message to DEC-writer
0F	D 0	10			10	
	1	0C			0C	
0D	D 2	21	VALUE	LPI	HL	Load starting address where MC reading is
	3	56			56	stored in actual form
	4	30			30	
	5	11		LPI	DE	Load starting address where MC reading is to be
	6	03			03	stored in ASCII form
	7	31			31	
	8	CD		JSR	UN(ACT-ASCII)	Convert data from actual form into ASCII
	9	C3			C3	form
	A	00			00	
	B	21		LPI	HL	Load starting address where MC reading is
	C	03			03	stored in ASCII
	D	31			31	
	E	CD		JSR	UN(BUFOUT)	Output MC reading to DEC-writer
	F	10			10	

HEXADECIMAL PAGE ADR	LINE ADR	INSTR	MNEMONIC LABEL	INSTR	MODIFIER	TITLE / DATE COMMENTS
0F	E 0	0C			0C	
	1	C3		JMP	UN(UNLOAD-GRAIN)	Jump to routine in main program to
	2	01			01	unload grain from test cell
	3	08			08	
0F	E 4	21	HI-INDIV-MESS	LPI	HL	Load starting address where message stating
	5	21			21	indiv MC reading is too high is stored
	6	12			12	
	7	CD		JSR	UN(BUFOUT)	Output message to DEC-writer
	8	10			10	
	9	0C			0C	
	A	C9		RET	UN	Return to calling routine
0F	E B	21	LO-INDIV-MESS	LPI	HL	Load starting address where message stating
	C	53			53	indiv MC reading is too low is stored
	D	12			12	
	E	CD		JSR	UN(BUFOUT)	Output message to DEC-writer
	F	10			10	
0F	F 0	0C			0C	
	1	C9		RET	UN	Return to calling routine
0F	F 2	21	HI-AVG-MESS	LPI	HL	Load starting address where message stating
	3	84			84	avg MS is too high is stored
	4	12			12	
	5	CD		JSR	UN(BUFOUT)	Output message to DEC-writer
	6	10			10	
	7	0C			0C	
	8	C9		RET	UN	Return to calling routine
0F	F 9	21	LO-[illegible]G-MESS	LPI	HL	Load starting address where message stating
	A	AB			AB	avg MC is too low is stored
	B	12			12	
	C	CD		JSR	UN(BUFOUT)	Output message to DEC-writer
	D	10			10	
	E	0C			0C	
	F	C9		RET	UN	Return to calling routine

HEXADECIMAL PAGE ADR	LINE ADR	INSTR	LABEL	MNEMONIC INSTR	MODIFIER	TITLE / COMMENTS DATE
15	8 0	0F		NOP		Calculator instructions, Calculator NOPS
	1	0F		NOP		
	2	2F		MCLR		Clear all calculator registers, MDC=8, floating point mode
	3	00		NOP		
	4	00		NOP		Reset on minute interrupt
15	8 5	3E	INTR-RESET	LDA	I	Output control word to control word register (30) is
	6	30			30	select counter 0 read/load LS byte first then
	7	D3		OUT		MS byte, interrupt on terminal count,
	8	27			27	Binary counter (16 bits)
	9	3E		LDA	I	
	A	70			70	Send to counter 0 LS byte first then
	B	D3		OUT		MS byte, count of 1770
	C	24			24	
	D	3E		LDA	I	
	E	17			17	
	F	D3		OUT		
15	9 0	24			24	
	1	C1		PLP	BC	Pull address to return after interrupt off stack
	2	C1		PLP	BC	Pull subroutine return address off stack
	3	FB		EIN		Enable interrupt
	4	C1		PLP	BC	Pull top two values off stack and push
	5	F1		PLP	AF	back on to have mode indicator in accumulator
	6	F5		PSP	AF	
	7	C5		PSP	BC	
	8	FE		CPA	I	Check if system is in wait mode (W)
	9	57			57	
	A	CA		JMP	ZI(TIMER)	If in wait mode jump to timer routine to
	B	1A			1A	update time
	C	08			08	
	D	C3		JMP	UN(LOAD-GRAIN)	If not in wait mode jump to routine to
	E	4B			4B	load grain into test cell
	F	09			09	

APPENDIX C

INSTRUCTION SET FOR THE 90006 CALCULATOR CARD

This appendix shows the instruction set for the Micro-Link 90006 calculator card which was used to perform calculations required by the automated moisture monitoring system.

MICRO LINK CORPORATION

Application Note 15
Calculator 90006

INTRODUCTION:

In many microprocessor based systems, the need to evaluate elaborate mathmatical equations often arises. Solution of these equations in software is difficult and time consuming! In order to relieve these difficulties, the calculator card (90006) was developed. With this card, complicated scientific equations can be solved with key level software similar to that used for a hand held RPN calculator. The 90006 Calculator Card is used as a peripheral to the central processor with Hand Shaking available to speed and simplify its use.

Because of the inherently slow nature of Number Crunching devices, parallel processing has been incorporated into the calculator card. Up to 64 instructions can be loaded into the calculator at speeds as great as one instruction per 400 nano-seconds.

POWER REQUIREMENTS:

The 90006 Calculator is intended for use with a split supply of + 5 volts and —10 volts at 200 ma and 180 ma respectively.

RESET:

The RESET input is an active low (0 volts) input which clears all memory buffers and registers on the calculator card. For proper operation, this line must be held low for at least 25 us.

After a RESET, the first two (2) instructions are not used, but must be present to initialize the calculator. Since these two instructions are not used, there value may be any possible combination of the six bit instruction word, NOP of ∅F (HEXADECIMAL) is suggested however.

READY:

The READY flag is an active high (+ 5 volts) signal which indicates when the input memory buffer has at least 1 of its cells empty and is therefore ready to receive the next instruction word. This line will most readily be used when a sequence of more than 64 instructions are needed. It must be monitored after the input buffer is filled since instructions cannot be entered faster than the number crunching unit can process them after this point. Whenever the STROBE-IN input is high, the READY output will go low and remain low until the STROBE-IN line is taken low again. If a memory cell is empty and ready to receive an instruction, the READY flag line will return to the high (+ 5 volts) state.

DATA AVAILABLE:

The DATA AVAILABLE flag is an active low (0 volts) signal which indicates if the data on the data output lines is valid. This line will go high (+ 5 volts) when the STROBE-OUT line is sent high and low again after STROBE-OUT is sent low and when another word is present in the output memory buffer.

ERROR:

The ERROR flag is an active low (0 volts) line used to indicate that an invalid instruction sequence such as ln (-2) or output instruction was processed, see Figure #1.

Figure #1

ERROR CONDITIONS

1. ln X } when X ≤ ∅
2. log X }
3. Any result < 10^{-99}
4. Any result > 10^{100}
5. TAN 90°, 270°, 450°, etc.
6. SIN X, COS X, TAN X, when |X| ≥ 9000°
7. SIN^{-1} X, COS^{-1} X when |X| > 1 or |X| ≤ 10^{-50}
8. $\sqrt{X}$ when X < 0
9. /, DIV/, 1/X when X = ∅
10. In floating point mode OUT instruction if number of mantissa digits to left of decimal point is > Mantissa Digit Count.

DATA:

These are the 4 lines, positive logic (active + 5 volts), which provide the result of an instruction sequence. The sequence of this data for both floating point and exponential notation is shown below:

Figure #2 - OUTPUT-FLOATING POINT

DATA WORD	DP POS	DO4	DO3	DO2	DO1
1		Sm	∅	∅	∅
2		DP POS			
3	11	Most significant Mantissa Digit = 0-9			
4	10	.			
.	.	.			
.	.	.			
.	.	.			
MDC + 2	12-MDC	Least significant Mantissa Digit = 0-9			

OUTPUT-EXPONENTIAL NOTATION

DATA WORD	DO4	DO3	DO2	DO1
1	Most significant exponent digit			
2	Least significant exponent digit			
3	Sm	∅	∅	Se
4	Not used			
5	Most significant mantissa digit (Decimal point follows this digit)			
.	.			
.	.			
.	.			
MDC + 4	Least significant mantissa digit			

MDC = Mantissa digit count, set by SMDC instruction, initially = 8

Sm = Sign of mantissa, 0 = positive, 1 = neg.

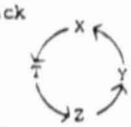
Se = Sign of exponent (Se = 0 in floating point mode)

DP POS = Decimal point position indicator is a value in the range from 11 down to 12-MDC, which indicates a digit, as given by the DP POS column in the table. The decimal point is located to the right of this digit.

The STROBE-IN is an active high (+ 5 volts) input used to write instructions into the input memory buffer. An onboard delay of STROBE-IN (25 ns) will allow the writing of both instruction and STROBE-IN to the same latched port at the same time for many ports such as the 74175 latch. See the sample program for methods of writing to the 90006 from a 4-bit microprocessor.

The STROBE-OUT is an active high (+ 5 volts) input used to clear the data lines. When STROBE-OUT is sent high, DATA AVAILABLE will go inactive (high). When STROBE-OUT is returned low, DATA AVAILABLE will return active if another word is present in the output memory buffer. Data transfer occurs on the negative edge of STROBE-OUT.

INSTRUCTION SET

CLASS	MNEMONIC*	HEXADECIMAL OP CODE	FULL NAME	DESCRIPTION
Digit Entry	0	ØØ	0	Mantissa or exponent digits. On first digit (d) the following occurs: Z → T Y → Z X → Y d → X
	1	Ø1	1	
	2	Ø2	2	
	3	Ø3	3	
	4	Ø4	4	
	5	Ø5	5	
	6	Ø6	6	
	7	Ø7	7	
	8	Ø8	8	
	9	Ø9	9	
	DP	ØA	Decimal Point	Digits that follow will be mantissa fraction
	EE	ØB	Enter Exponent	Digits that follow will be exponent
	CS	ØC	Change Sign	Change sign of exponent or mantissa Xm = X mantissa Xe = X exponent CS causes - Xm → Xm or - Xe → Xe depending on whether or not an EE instruction was executed after last number entry initiation
	PI	ØD	Constant	3.1415927 → X, stack not pushed.
	EN	21	Enter	Terminates digit entry and pushes the stack The argument entered will be in X and Y Z → T Y → Z X → Y
	NOP	ØF	No Operation	Do Nothing Instruction
Move	ROLL	23	Roll	Roll Stack 
	POP	2E	Pop	Pop Stack Y → X Z → Y T → Z 0 → T
	XEY	30	X exchange Y	Exchange X and Y X ↔ Y
	XEM	1B	X exchange M	Exchange X with Memory X ↔ M
	MS	1C	Memory Store	Store X in Memory X ↔ M
	MR	1D	Memory Recall	Recall Memory into X M ↔ X
	LSH	1E	Left Shift Xm	X mantissa is left shifted while leaving decimal point in same position. Former most significant digit is saved in link digit. Least significant digit is zero.
	RSH	1F	Right Shift Xm	X mantissa is right shifted while leaving decimal point is same position. Link digit, which is normally zero except after a left shift, is shifted into the most significant digit. Least significant digit is lost.

ORIGINAL PAGE IS
OF POOR QUALITY

Math	+	39	Plus	Add X to Y. X + Y ↔ X. On +, -, X, / and YX instructions, stack is popped as follows: Z → Y, T → Z, O → T. Former X, Y are lost
	-	3A	Minus	Subtract X from Y, Y - X X → X
	x	3B	Times	Multiply X times Y. Y x X → X
	/	3C	Divide	Divide X into Y. Y / X → X
	YX	38	Y to X	Raise Y to X power. $Y^X \to X$
	INV +*	20, 39	Memory Plus	Add X to memory. M + X → M. On INV +, -, x and / instructions, X, Y, Z, and T are unchanged.
	INV-*	20, 3A	Memory Minus	Subtract X from memory. M - X → M
	INV x*	20, 3B	Memory Times	Multiply X times memory. M x X → M
	INV /*	20, 3C	Memory Divide	Divide X into memory. M / X → M
	1/X	37	One Divided by X	1 / X → X. On all F (X) math instructions Y, Z, T and M are unchanged and previous X is lost.
	SQRT	34	Square Root	$\sqrt{X} \to X$
	SQ	33	Square	$X^2 \to X$
	10X	32	Ten to X	$10^X \to X$
	EX	31	E to X	$e^X \to X$
	LN	35	Natural log of X	ln X → X
	LOG	36	Base 10 log of X	log X → X
	SIN	24	Sine X	SIN (X) X. On all F(X) trig functions Y, Z, T and M are unchanged and the previous X is lost.
	COS	25	Cosine X	COS (X) → X
	TAN	26	Tangent X	TAN (X) → X
	INV SIN*	20, 24	Inverse sine X	SIN^{-1} (X) → X
	INV COS*	20, 25	Inverse cosine X	COS^{-1} (X) → X
	INV TAN*	20, 26	Inverse tan X	TAN^{-1} (X) → X
	DTR	2D	Degrees to radians	Convert X from degress to radians.
	RTD	2C	Radians to degrees	Convert X from radians to degrees.
Clear	MCLR	2F	Master Clear	Clear all internal registers and memory; initialize out control signals, MDC = 8, MODE = floating point.
	ECLR	2B	Error Flag Clear	1 Error Flag
Memory	IBNZ	19, xx**	Increment memory	M + 1 → M
	DBNZ	1A, xx**	Decrement memory	M - 1 → M
Output	OUT*	16, xx	Multidigit output from X	Writes the contents of the x register to the output memory buffer. After an "out" data available will become active.
Mode Control	TOGM	22	Toggle Mode	Change mode from floating point to scientific notation or vice-versa, depending on present mode. The mode affects only the IN and OUT instructions. Internal calculations are always in 8-digit scientific notation.
	SMDC*	18	Set Mantissa Digit Count	Mantissa digit count is set to the contents of the second instruction word (=1 to 8).
	INV	20	Inverse Mode	Set inverse mode for trig or memory function instruction that will immediately follow. Inverse mode is for next instruction only.

NOTES: The two most significant bits of the instruction words are never used. See the block diagram on the front of the data sheet.

* 2 - Word Instruction

** X = Don't Care

APPENDIX D

PRINTOUTS FROM THE PERFORMANCE TEST

Appendix D contains the printouts from the performance test. This includes one printout from performance test for incoming grain which is being unloaded, and a printout from performance test for outbound grain being loaded for shipment.

Incoming grain performance test

```
WHAT IS THE DATE?
09-19-80
WHAT TIME WILL THE READINGS BEGIN?
03:10 PM
WHAT IS THE UPPER LIMIT FOR AN INDIVIDUAL MOISTURE CONTENT READING?
17.5
WHAT IS THE LOWER LIMIT FOR AN INDIVIDUAL MOISTURE CONTENT READING?
14.5
WHAT IS THE UPPER LIMIT FOR AVERAGE MOISTURE CONTENT?
16.5
WHAT IS THE LOWER LIMIT FOR AVERAGE MOISTURE CONTENT?
15.5
IS GRAIN BEING UNLOADED?
Y
WHAT BIN IS GRAIN GOING INTO?
# 55


  DATE        TIME      INDIV MC(%wb)   AVG MC(%wb)   GRAIN DESTINATION
09-19-80   03:10 PM         16.5            16.5           # 55
09-19-80   03:11 PM         16.2            16.4           # 55
09-19-80   03:12 PM         16.3            16.3           # 55

EXCESSIVELY LOW READING, IGNORED  00.1

EXCESSIVELY HIGH READING, IGNORED  98.9
09-19-80   03:15 PM         17.5            16.6           # 55
AVERAGE MOISTURE CONTENT IS TOO HIGH
09-19-80   03:16 PM         15.3            16.4           # 55
09-19-80   03:17 PM         16.1            16.3           # 55
09-19-80   03:18 PM         17.5            16.5           # 55
09-19-80   03:19 PM         17.2            16.6           # 55
AVERAGE MOISTURE CONTENT IS TOO HIGH
09-19-80   03:20 PM         15.3            16.4           # 55
09-19-80   03:21 PM         15.9            16.4           # 55
09-19-80   03:22 PM         17.9            16.5           # 55
INDIVIDUAL MOISTURE CONTENT READING IS TOO HIGH
09-19-80   03:23 PM         16.0            16.5           # 55
09-19-80   03:24 PM         16.3            16.5           # 55
09-19-80   03:25 PM         15.2            16.4           # 55
```

```
INDIVIDUAL MOISTURE CONTENT READING IS TOO LOW
09-19-80   03:27 PM      17.3      16.2      # 55
09-19-80   03:28 PM      15.1      16.1      # 55
09-19-80   03:29 PM      16.0      16.1      # 55
09-19-80   03:30 PM      15.2      16.1      # 55
09-19-80   03:31 PM      16.2      16.1      # 55
09-19-80   03:32 PM      15.0      16.0      # 55
09-19-80   03:33 PM      17.5      16.1      # 55
09-19-80   03:34 PM      15.9      16.1      # 55
09-19-80   03:35 PM      17.4      16.2      # 55
09-19-80   03:36 PM      15.3      16.1      # 55
09-19-80   03:37 PM      14.9      16.1      # 55
09-19-80   03:38 PM      17.3      16.1      # 55
09-19-80   03:39 PM      15.4      16.1      # 55
09-19-80   03:40 PM      16.2      16.1      # 55
09-19-80   03:41 PM      15.0      16.1      # 55
09-19-80   03:42 PM      16.3      16.1      # 55
09-19-80   03:43 PM      15.0      16.0      # 55
09-19-80   03:44 PM      17.2      16.1      # 55
09-19-80   03:45 PM      15.2      16.0      # 55
09-19-80   03:46 PM      17.3      16.1      # 55
09-19-80   03:47 PM      17.4      16.1      # 55
09-19-80   03:48 PM      16.4      16.1      # 55
09-19-80   03:49 PM      15.2      16.1      # 55
09-19-80   03:50 PM      16.3      16.1      # 55
09-19-80   03:51 PM      15.1      16.1      # 55
09-19-80   03:52 PM      15.0      16.1      # 55
09-19-80   03:53 PM      16.0      16.1      # 55
09-19-80   03:54 PM      15.0      16.0      # 55
09-19-80   03:55 PM      16.0      16.0      # 55
09-19-80   03:56 PM      15.4      16.0      # 55
09-19-80   03:57 PM      17.4      16.0      # 55
09-19-80   03:58 PM      16.4      16.1      # 55
09-19-80   03:59 PM      15.0      16.0      # 55
09-19-80   04:00 PM      16.5      16.0      # 55
09-19-80   04:01 PM      15.9      16.0      # 55
09-19-80   04:02 PM      17.3      16.1      # 55
09-19-80   04:03 PM      16.5      16.1      # 55
09-19-80   04:04 PM      12.8      16.0      # 55
INDIVIDUAL MOISTURE CONTENT READING IS TOO LOW
09-19-80   04:05 PM      12.7      15.9      # 55
INDIVIDUAL MOISTURE CONTENT READING IS TOO LOW
09-19-80   04:06 PM      15.0      15.9      # 55                W

WAIT MODE
          R

RUN MODE

09-19-80   04:30 PM      16.0      15.9      # 55
```

09-19-80	04:32 PM	14.9	15.9	# 55
09-19-80	04:33 PM	17.1	16.0	# 55
09-19-80	04:34 PM	17.0	16.0	# 55
09-19-80	04:35 PM	16.2	16.0	# 55
09-19-80	04:36 PM	17.0	16.0	# 55
09-19-80	04:37 PM	14.8	16.0	# 55
09-19-80	04:38 PM	18.9	16.0	# 55
INDIVIDUAL MOISTURE CONTENT READING IS TOO HIGH				
09-19-80	04:39 PM	15.0	16.0	# 55
09-19-80	04:40 PM	15.0	16.0	# 55
09-19-80	04:41 PM	12.7	15.9	# 55
INDIVIDUAL MOISTURE CONTENT READING IS TOO LOW				
09-19-80	04:42 PM	17.2	16.0	# 55
09-19-80	04:43 PM	17.2	16.0	# 55
09-19-80	04:44 PM	18.8	16.0	# 55
INDIVIDUAL MOISTURE CONTENT READING IS TOO HIGH				
09-19-80	04:45 PM	15.0	16.0	# 55
09-19-80	04:46 PM	16.1	16.0	# 55
09-19-80	04:47 PM	17.1	16.0	# 55
09-19-80	04:48 PM	15.1	16.0	# 55
09-19-80	04:49 PM	16.2	16.0	# 55
09-19-80	04:50 PM	15.1	16.0	# 55
09-19-80	04:51 PM	14.8	16.0	# 55
09-19-80	04:52 PM	16.6	16.0	# 55
09-19-80	04:53 PM	16.4	16.0	# 55
09-19-80	04:54 PM	13.0	16.0	# 55
INDIVIDUAL MOISTURE CONTENT READING IS TOO LOW				
09-19-80	04:55 PM	17.3	16.0	# 55
09-19-80	04:56 PM	12.9	15.9	# 55
INDIVIDUAL MOISTURE CONTENT READING IS TOO LOW				
09-19-80	04:57 PM	17.5	16.0	# 55
09-19-80	04:58 PM	16.1	16.0	# 55
09-19-80	04:59 PM	17.1	16.0	# 55
09-19-80	05:00 PM	15.3	16.0	# 55
09-19-80	05:01 PM	15.1	15.9	# 55
09-19-80	05:02 PM	16.4	16.0	# 55
09-19-80	05:03 PM	17.3	16.0	# 55
09-19-80	05:04 PM	17.2	16.0	# 55
09-19-80	05:05 PM	15.1	16.0	# 55
09-19-80	05:06 PM	15.1	16.0	# 55
09-19-80	05:07 PM	15.1	16.0	# 55
09-19-80	05:08 PM	17.2	16.0	# 55
09-19-80	05:09 PM	18.9	16.0	# 55
INDIVIDUAL MOISTURE CONTENT READING IS TOO HIGH				
09-19-80	05:10 PM	15.0	16.0	# 55
09-19-80	05:11 PM	16.5	16.0	# 55
09-19-80	05:12 PM	16.2	16.0	# 55
09-19-80	05:13 PM	16.1	16.0	# 55
09-19-80	05:14 PM	17.3	16.0	# 55
09-19-80	05:15 PM	17.1	16.0	# 55

09-19-80	05:17 PM	16.4	16.0	# 55
09-19-80	05:18 PM	16.8	16.0	# 55
09-19-80	05:19 PM	15.0	16.0	# 55
09-19-80	05:20 PM	16.7	16.0	# 55
09-19-80	05:21 PM	17.1	16.0	# 55
09-19-80	05:22 PM	15.0	16.0	# 55
09-19-80	05:23 PM	16.9	16.0	# 55
09-19-80	05:24 PM	18.5	16.1	# 55
INDIVIDUAL MOISTURE CONTENT READING IS TOO HIGH				
09-19-80	05:25 PM	16.2	16.1	# 55
09-19-80	05:26 PM	16.0	16.1	# 55
09-19-80	05:27 PM	16.0	16.1	# 55
09-19-80	05:28 PM	19.1	16.1	# 55
INDIVIDUAL MOISTURE CONTENT READING IS TOO HIGH				
09-19-80	05:29 PM	18.9	16.1	# 55
INDIVIDUAL MOISTURE CONTENT READING IS TOO HIGH				
09-19-80	05:30 PM	17.1	16.1	# 55
09-19-80	05:31 PM	18.9	16.1	# 55
INDIVIDUAL MOISTURE CONTENT READING IS TOO HIGH				
09-19-80	05:32 PM	16.1	16.1	# 55
09-19-80	05:33 PM	16.1	16.1	# 55
09-19-80	05:34 PM	17.2	16.2	# 55

Outgoing grain performance test

WHAT IS THE DATE?
09-19-80
WHAT TIME WILL THE READINGS BEGIN?
07:10 PM
WHAT IS THE UPPER LIMIT FOR AN INDIVIDUAL MOISTURE CONTENT READING?
16.2
WHAT IS THE LOWER LIMIT FOR AN INDIVIDUAL MOISTURE CONTENT READING?
14.8
WHAT IS THE UPPER LIMIT FOR AVERAGE MOISTURE CONTENT?
15.8
WHAT IS THE LOWER LIMIT FOR AVERAGE MOISTURE CONTENT?
15.0
IS GRAIN BEING UNLOADED?
NO
WHAT IS GRAIN BEING LOADED ONTO?
BARGE # 20
APPROXIMATELY HOW MANY READINGS WILL IT TAKE TO LOAD THE GRAIN?
120.0
WHAT IS THE DESIRED FINAL AVERAGE MOISTURE CONTENT?
15.4
HOW MANY READINGS SHOULD PASS BETWEEN EACH CALCULATION OF DESIRED MOISTURE CONTENT FOR THE REMAINDER OF THE GRAIN?
10

DATE	TIME	INDIV MC(%wb)	AVG MC(%wb)	GRAIN DESTINATION
09-19-80	07:10 PM	15.0	15.0	BARGE # 20
09-19-80	07:11 PM	15.3	15.2	BARGE # 20
09-19-80	07:12 PM	14.9	15.1	BARGE # 20
09-19-80	07:13 PM	14.9	15.0	BARGE # 20
09-19-80	07:14 PM	15.0	15.0	BARGE # 20
09-19-80	07:15 PM	15.1	15.0	BARGE # 20
09-19-80	07:16 PM	14.9	15.0	BARGE # 20
09-19-80	07:17 PM	15.1	15.0	BARGE # 20
09-19-80	07:18 PM	15.0	15.0	BARGE # 20
09-19-80	07:19 PM	15.1	15.0	BARGE # 20

THE REMAINDER OF THE GRAIN SHOULD HAVE AN AVERAGE MC PERCENT OF 15.4

09-19-80	07:20 PM	15.1	15.0	BARGE # 20
09-19-80	07:21 PM	15.2	15.1	BARGE # 20
09-19-80	07:22 PM	14.7	15.0	BARGE # 20

INDIVIDUAL MOISTURE CONTENT READING IS TOO LOW

09-19-80	07:23 PM	15.0	15.0	BARGE # 20

09-19-80	07:24 PM	15.1	15.0	BARGE # 20
09-19-80	07:25 PM	14.8	15.0	BARGE # 20
09-19-80	07:26 PM	15.0	15.0	BARGE # 20
09-19-80	07:27 PM	14.9	15.0	BARGE # 20
09-19-80	07:28 PM	15.0	15.0	BARGE # 20
09-19-80	07:29 PM	14.9	15.0	BARGE # 20

THE REMAINDER OF THE GRAIN SHOULD HAVE AN AVERAGE MC PERCENT OF 15.5

09-19-80	07:30 PM	14.9	15.0	BARGE # 20
09-19-80	07:31 PM	14.6	15.0	BARGE # 20

INDIVIDUAL MOISTURE CONTENT READING IS TOO LOW

09-19-80	07:32 PM	14.9	15.0	BARGE # 20
09-19-80	07:33 PM	14.9	15.0	BARGE # 20
09-19-80	07:34 PM	14.8	15.0	BARGE # 20
09-19-80	07:35 PM	15.1	15.0	BARGE # 20
09-19-80	07:36 PM	14.7	15.0	BARGE # 20

INDIVIDUAL MOISTURE CONTENT READING IS TOO LOW

09-19-80	07:37 PM	14.9	15.0	BARGE # 20
09-19-80	07:38 PM	15.2	15.0	BARGE # 20
09-19-80	07:39 PM	14.9	15.0	BARGE # 20

THE REMAINDER OF THE GRAIN SHOULD HAVE AN AVERAGE MC PERCENT OF 15.5

09-19-80	07:40 PM	15.1	15.0	BARGE # 20
09-19-80	07:41 PM	15.0	15.0	BARGE # 20
09-19-80	07:42 PM	14.8	15.0	BARGE # 20
09-19-80	07:43 PM	15.1	15.0	BARGE # 20
09-19-80	07:44 PM	14.8	15.0	BARGE # 20
09-19-80	07:45 PM	15.0	15.0	BARGE # 20
09-19-80	07:46 PM	15.0	15.0	BARGE # 20
09-19-80	07:47 PM	15.3	15.0	BARGE # 20
09-19-80	07:48 PM	14.9	15.0	BARGE # 20
09-19-80	07:49 PM	15.1	15.0	BARGE # 20

THE REMAINDER OF THE GRAIN SHOULD HAVE AN AVERAGE MC PERCENT OF 15.6

09-19-80	07:50 PM	14.9	15.0	BARGE # 20
09-19-80	07:51 PM	14.9	15.0	BARGE # 20
09-19-80	07:52 PM	14.8	15.0	BARGE # 20
09-19-80	07:53 PM	14.9	15.0	BARGE # 20
09-19-80	07:54 PM	14.9	15.0	BARGE # 20
09-19-80	07:55 PM	14.8	15.0	BARGE # 20
09-19-80	07:56 PM	14.7	15.0	BARGE # 20

INDIVIDUAL MOISTURE CONTENT READING IS TOO LOW

09-19-80	07:57 PM	15.2	15.0	BARGE # 20
09-19-80	07:58 PM	14.8	15.0	BARGE # 20
09-19-80	07:59 PM	14.8	15.0	BARGE # 20

THE REMAINDER OF THE GRAIN SHOULD HAVE AN AVERAGE MC PERCENT OF 15.7

```
09-19-80   08:00 PM      15.2         15.0       BARGE # 20
09-19-80   08:01 PM      15.0         15.0       BARGE # 20
09-19-80   08:02 PM      15.0         15.0       BARGE # 20
09-19-80   08:03 PM      14.8         15.0       BARGE # 20
09-19-80   08:04 PM      15.0         15.0       BARGE # 20
09-19-80   08:05 PM      15.1         15.0       BARGE # 20
09-19-80   08:06 PM      14.8         15.0       BARGE # 20
09-19-80   08:07 PM      15.0         15.0       BARGE # 20
09-19-80   08:08 PM      14.9         15.0       BARGE # 20
09-19-80   08:09 PM      14.7         15.0       BARGE # 20
INDIVIDUAL MOISTURE CONTENT READING IS TOO LOW

THE REMAINDER OF THE GRAIN SHOULD HAVE AN AVERAGE MC PERCENT OF 15.8

09-19-80   08:10 PM      14.8         15.0       BARGE # 20
09-19-80   08:11 PM      14.9         15.0       BARGE # 20
09-19-80   08:12 PM      15.0         15.0       BARGE # 20
09-19-80   08:13 PM      14.9         15.0       BARGE # 20
09-19-80   08:14 PM      15.0         15.0       BARGE # 20
09-19-80   08:15 PM      14.7         14.9       BARGE # 20
INDIVIDUAL MOISTURE CONTENT READING IS TOO LOW
AVERAGE MOISTURE CONTENT IS TOO LOW
09-19-80   08:16 PM      14.8         14.9       BARGE # 20
AVERAGE MOISTURE CONTENT IS TOO LOW
09-19-80   08:17 PM      14.8         14.9       BARGE # 20
AVERAGE MOISTURE CONTENT IS TOO LOW
09-19-80   08:18 PM      14.6         14.9       BARGE # 20
INDIVIDUAL MOISTURE CONTENT READING IS TOO LOW
AVERAGE MOISTURE CONTENT IS TOO LOW
09-19-80   08:19 PM      14.8         14.9       BARGE # 20
AVERAGE MOISTURE CONTENT IS TOO LOW

THE REMAINDER OF THE GRAIN SHOULD HAVE AN AVERAGE MC PERCENT OF 16.1
                                                                      W

WAIT MODE
         R

RUN MODE

09-19-80   08:22 PM      16.0         15.0       BARGE # 20
09-19-80   08:23 PM      13.2         14.9       BARGE # 20
INDIVIDUAL MOISTURE CONTENT READING IS TOO LOW
AVERAGE MOISTURE CONTENT IS TOO LOW
09-19-80   08:24 PM      16.3         14.9       BARGE # 20
INDIVIDUAL MOISTURE CONTENT READING IS TOO HIGH
AVERAGE MOISTURE CONTENT IS TOO LOW
09-19-80   08:25 PM      16.0         15.0       BARGE # 20
09-19-80   08:26 PM      16.2         15.0       BARGE # 20
09-19-80   08:27 PM      16.1         15.0       BARGE # 20
09-19-80   08:28 PM      16.3         15.0       BARGE # 20
```

09-19-80	08:29 PM	16.1	15.0	BARGE # 20
09-19-80	08:30 PM	16.1	15.0	BARGE # 20
09-19-80	08:31 PM	16.2	15.1	BARGE # 20

THE REMAINDER OF THE GRAIN SHOULD HAVE AN AVERAGE MC PERCENT OF 16.1

09-19-80	08:32 PM	16.2	15.1	BARGE # 20
09-19-80	08:33 PM	16.1	15.1	BARGE # 20
09-19-80	08:34 PM	16.3	15.1	BARGE # 20

INDIVIDUAL MOISTURE CONTENT READING IS TOO HIGH

09-19-80	08:35 PM	16.2	15.1	BARGE # 20
09-19-80	08:36 PM	15.9	15.1	BARGE # 20
09-19-80	08:37 PM	16.0	15.1	BARGE # 20
09-19-80	08:38 PM	16.0	15.1	BARGE # 20
09-19-80	08:39 PM	16.5	15.2	BARGE # 20

INDIVIDUAL MOISTURE CONTENT READING IS TOO HIGH

09-19-80	08:40 PM	15.9	15.2	BARGE # 20
09-19-80	08:41 PM	15.9	15.2	BARGE # 20

THE REMAINDER OF THE GRAIN SHOULD HAVE AN AVERAGE MC PERCENT OF 16.1

09-19-80	08:42 PM	15.9	15.2	BARGE # 20
09-19-80	08:43 PM	16.3	15.2	BARGE # 20

INDIVIDUAL MOISTURE CONTENT READING IS TOO HIGH

09-19-80	08:44 PM	16.0	15.2	BARGE # 20
09-19-80	08:45 PM	15.9	15.2	BARGE # 20
09-19-80	08:46 PM	15.9	15.2	BARGE # 20
09-19-80	08:47 PM	15.9	15.2	BARGE # 20
09-19-80	08:48 PM	16.0	15.2	BARGE # 20
09-19-80	08:49 PM	16.0	15.2	BARGE # 20
09-19-80	08:50 PM	16.4	15.2	BARGE # 20

INDIVIDUAL MOISTURE CONTENT READING IS TOO HIGH

09-19-80	08:51 PM	16.1	15.3	BARGE # 20

THE REMAINDER OF THE GRAIN SHOULD HAVE AN AVERAGE MC PERCENT OF 16.[illegible]

09-19-80	08:52 PM	16.0	15.3	BARGE # 20
09-19-80	08:53 PM	16.0	15.3	BARGE # 20
09-19-80	08:54 PM	16.0	15.3	BARGE # 20
09-19-80	08:55 PM	15.6	15.3	BARGE # 20
09-19-80	08:56 PM	16.0	15.3	BARGE # 20
09-19-80	08:57 PM	16.0	15.3	BARGE # 20
09-19-80	08:58 PM	16.0	15.3	BARGE # 20
09-19-80	08:59 PM	16.2	15.3	BARGE # 20
09-19-80	09:00 PM	16.1	15.3	BARGE # 20.
09-19-80	09:01 PM	16.0	15.3	BARGE # 20

THE REMAINDER OF THE GRAIN SHOULD HAVE AN AVERAGE MC PERCENT OF 16.3

09-19-80	09:02 PM	15.8	15.3	BARGE # 20

09-19-80	09:04 PM	16.1	15.3	BARGE # 20
09-19-80	09:05 PM	16.1	15.3	BARGE # 20
09-19-80	09:06 PM	15.8	15.3	BARGE # 20
09-19-80	09:07 PM	16.4	15.4	BARGE # 20
INDIVIDUAL MOISTURE CONTENT READING IS TOO HIGH				
09-19-80	09:08 PM	15.9	15.4	BARGE # 20
09-19-80	09:09 PM	16.0	15.4	BARGE # 20
09-19-80	09:10 PM	16.0	15.4	BARGE # 20
09-19-80	09:11 PM	15.9	15.4	BARGE # 20

THE PREDICTED NUMBER OF READINGS HAVE BEEN TAKEN.

09-19-80	09:12 PM	16.3	15.4	BARGE # 20
INDIVIDUAL MOISTURE CONTENT READING IS TOO HIGH				
09-19-80	09:13 PM	16.5	15.4	BARGE # 20
INDIVIDUAL MOISTURE CONTENT READING IS TOO HIGH				
09-19-80	09:14 PM	16.0	15.4	BARGE # 20
09-19-80	09:15 PM	16.0	15.4	BARGE # 20
09-19-80	09:16 PM	16.0	15.4	BARGE # 20
09-19-80	09:17 PM	15.8	15.4	BARGE # 20
09-19-80	09:18 PM	15.8	15.4	BARGE # 20
09-19-80	09:19 PM	16.0	15.4	BARGE # 20
09-19-80	09:20 PM	15.8	15.4	BARGE # 20
09-19-80	09:21 PM	15.8	15.4	BARGE # 20
09-19-80	09:22 PM	15.8	15.4	BARGE # 20
09-19-80	09:23 PM	15.8	15.4	BARGE # 20
09-19-80	09:24 PM	15.8	15.4	BARGE # 20
09-19-80	09:25 PM	16.1	15.4	BARGE # 20